AN ABSTRACT OF THE THESIS OF

Dedrie Beardsley for the degree of Master of Arts in Interdisciplinary Studies in Computer Science, Psychology and Psychology presented on June 7th, 2013.

Title: Inspiring End-user Programming Through Exposure to Multi-platform Computing and Improved Integrated Development Environments

Abstract approved:

_____

Carlos Jensen

The art of software engineering inherently requires high-level problem solving and perseverance, as programmers and designers wrestle with complex design and implementation challenges in the process of turning loose concepts and ideas into working code. In the current developer ecosystem, engineers are commonly incentivized extrinsically by monetary rewards, approval or status. Ironically, extrinsically motivating a person has been proven to decrease complex problem solving performance. On the other hand, motivating a person intrinsically through the Self Deterministic Theory constructs of autonomy, competence and relatedness has been shown to encourage problem solving, creativity and innovation. To determine what motivates different types of developers, and how well their tools support them in their work, we developed and deployed a survey. The validated survey tool measured the intrinsic motivation level of 103 developers of varying personas. Based on the data gathered, we

highlight areas for improving the current development environment to foster increased problem solving and creativity.

Inspiring End-user Programming Through Exposure to Multi-platform Computing and Improved Integrated Development Environments

by
Dedrie Beardsley

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Arts in Interdisciplinary Studies

Presented June 7, 2013
Commencement June 2014

Master of Arts thesis of Dedrie Beardsley presented on June 7,2013

APPROVED:

_____

Major Professor, representing Computer Science

_____

Director of the Interdisciplinary Studies Program

_____

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

_____

Dedrie Beardsley, Author

ACKNOWLEDGEMENTS

I would like to thank Intel Corporation for their generous financial support and mentorship; specifically Shannon Schroeder.

I would like to thank my advisor Carlos Jensen for his guidance and assistance during my time at OSU. I would also like to thank several fellow research associates for their contributions. Without their assistance my projects would not have been possible; Justin Wolford, Dale Cox, Soroush Ghorashi and Rahul Gopinath.

I would like to thank my committee members Mei Lien, Sarina Saturn and Michele Swift for their guidance, time and encouragement.

Hal Koenig made a significant time contribution in survey design and statistical analysis guidance. Without his efforts, this study would not have been possible. I am extremely grateful for his assistance.

Finally, I would like to thank an OSU alumni and my cousin Eric Betts for his encouragement to attend graduate school, his technical assistance and his perpetual advocacy for myself and other women in the field of computer science.

CONTRIBUTION OF AUTHORS

Manuscript #1

I was primary author, experimental designer and conducted the study and analysis for the first manuscript. Rahul Gopinath contributed as a secondary author primarily in the related works section.

Manuscript#2

I helped design and conduct the experiment as well as data collection and analysis. For this manuscript Dale Cox was primary author, Justin Wolford was the secondary author contributing to the related work sections and qualitative analysis. I assisted with related works and general editing. Dale developed the applications used to interface with the Microsoft Kinect and Nintendo Wii Remote used during the research study. He also contributed to designing and conducting the research study. Justin Wolford helped design and conduct the research study, as well as analyzing the data collected during the experiment. He also developed the drag and drop application used as the first task.

TABLE OF CONTENTS

Page

LIST OF FIGURES

Figure Page

Manuscript #1

Manuscript #2

LIST OF TABLES

LIST OF APPENDIX TABLES

Figure Page

1. INTRODUCTION

We began this journey evaluating interactive television (iTV) environments from the perspective of an end-user. Through this process we learned that these users had many ideas for customization and optimization in iTV, but no means to implement them. It stands to reason that users experiencing platforms other than iTV would also have similar innovative ideas. There has been huge growth in ubiquitous computing interactions such as iTV, smart phones, tablets and in vehicle infotainment. Over 1 billion smart phones are currently being used worldwide and by 2015 this number is projected to double [1]. That is a substantial number of potential innovators experiencing the same deficit in appropriate mechanisms to develop their ideas. Skilled programmers use a software application that combines multiple development tools known as an integrated development environment (IDE) to innovate. However, very few IDEs have features allowing novices to create software solutions. When considering factors that support innovation such as intrinsic motivation, collaboration, flexibility and recognition, it becomes clear that modern IDEs aren't supporting innovation in skilled developers either. We examined the modern IDE in order to make recommendations for improving innovation support. We did this by measuring the basic psychological needs that support intrinsic motivation; autonomy, relatedness and competence in developers that use IDEs [2,3].

Self-Deterministic Theory in Integrated Development Environments: Methods to Enhance Problem Solving, Learning and Exploring in Developers

Dedrie Beardsley, Rahul Gopinath and Carlos Jensen

2. ABSTRACT

The art of software engineering inherently requires high-level problem solving and perseverance, as programmers and designers wrestle with complex design and implementation challenges in the process of turning loose concepts and ideas into working code. In the current developer ecosystem, engineers are commonly incentivized extrinsically by monetary rewards, approval or status. Ironically, extrinsically motivating a person has been proven to decrease complex problem solving performance. On the other hand, motivating a person intrinsically through the Self Deterministic Theory constructs of autonomy, competence and relatedness has been shown to encourage problem solving, creativity and innovation. To determine what motivates different types of developers, and how well their tools support them in their work, we developed and deployed a survey. The validated survey tool measured the intrinsic motivation level of 103 developers of varying personas. Based on the data gathered, we highlight areas for improving the current development environment to foster increased problem solving and creativity.

3. INTRODUCTION

It is commonly thought that the best predictors of engineer performance are experience and knowledge. However research has shown that motivational factors have a much larger influence [1,53]. These motivational factors drive software engineers to translate abstract concepts and requirements into concrete artifacts [7]. This process requires creating novel and unique ways to transform loose ideas into a product and is frequently mediated by a development environment. Accomplishing this task requires a complex iterative process and a large range of roles and capabilities.

According to Turley and Beiman, the most exceptional engineers share and employ motivational traits such as a bias toward action, a systems based perspective, a sense of mission, strong convictions and proactively helping other colleagues [53]. Additionally, sixty percent of the engineering student outcomes required by The Accreditation Board for Engineering and Technology (ABET) contain the motivational principles of problem solving, analytical thinking, teamwork, communication, curiosity and life-long learning [1].

In summary the most successful engineers are team players, proficient problem solvers and have an internal drive to innovate; they are highly intrinsically motivated. Surprisingly, these traits have more to do with motivational factors than the common assumptions of experience, depth of knowledge and detail orientation. One could theorize that providing a development tool that supports these motivational traits would lead to more successful engineers.

The software engineering process and roles have changed significantly since the first IDEs were created. As this evolution occurred, IDE functions have stayed relatively the same and overlook the motivational factors that enhance

engineering performance. It was out of necessity that software developers created their own integrated environments to work in which primarily focused on productivity. Engineers would use a collection of tools to create product and out of efficiency, began to combine them together.

In 1981 Osterweil proposed that these tool collections should have five properties; Capabilities spanning the entire range of activities needed to complete a software job, user friendliness, tight tool integration, internal reusability and the use of central database [34]. Osterweil interpreted these properties into an early prototype known as Toolpack. Toolpack was presented as a sound basis for programming support and packaged features such as a compiling system, a Fortran-intelligent editor, a formatter, a structurer (infers and emphasizes the underlying looping structure of a program), dynamic testing, validation and de debugging, static error detection and validation, static portability checking, documentation generation and a program transformer [35]. The range of activities he perceived as necessary to complete a software project did not include activities outside of low-level programming. User friendliness was addressed in the form of simple heuristics and did not address developer work or innovation style.

Today, developing innovative software solutions requires teamwork and a variety of roles. The developer's role has changed and greatly expands past fundamental coding to developing for multiple platforms, project management, testing, architecture, design, writing, User experience (UX), agile, prototyping, emulation, pair programming and open source software (OSS) development. Development teams frequently collaborate with subject matter experts, other programming teams and each other. Programmers also interact with communities of developers outside their organization obtaining skills, advice, code snippets, and general support.

A whole new category of developers have also emerged; end-users. These programmers should not be overlooked in creating IDEs as they have significant potential to innovate. Although end-user programmers have existed for many years, the introduction of ubiquitous computing is likely to increase their numbers. It is estimated that there was 12 million American workers engaged in end-user programming in 2012 [46]. This number doesn't include people programming outside of an occupation or locations external to the U.S. A report by Evans Data estimates there will be 20 million developers worldwide by 2015, which is a 25% increase from the current population and is based on the growing ubiquitous markets [36]. Just 6% of these programmers reside in the U.S. [33] If end-user programming follows this trend there could be 250 million end-user programmers world wide by 2015.

As more individuals are exposed to applications via the ever-expanding platforms of mobile phones, tablets, interactive television, in-vehicle infotainment, home automation and social media, they will be inspired to create software that improves their lives. Translating these ideas into product can be difficult because development tools are geared towards advanced users and do not resemble the environment in which their applications would be used. Excluding end-user programming needs from the existing developer tool set is drastically reducing the amount of diversity and ideas available to the software engineering pool. We need to meet both the technical and motivational needs of these end-user programmers to elicit the innovation in waiting ultimately fueling the ubiquitous computing software development ecosystem.

3.1 In order to fuel this innovation in both end-user programmers and traditional developers, we need to broaden our focus from standard productivity to include intrinsic motivation. Early integrated tool sets addressed the needs specific to those who developed them. They were geared toward individual coding, advanced users and allowed for little self-directed behavior in the coding

execution. These coders developed for desktop and limited form factors and allowances. Programmers today must create applications that span an immense range of devices and users. As IDEs have evolved, some modern developer roles and technical needs have been addressed but developer motivation and diversity has largely been ignored. It is well supported that innovation is fed by creativity, diversity, collaboration and intrinsic motivation [3,50,56]. Perhaps the modern day IDE is limiting the growth of intrinsically motivated, diverse developer populations and thus rapid innovation today. One way to improve on this would be to introduce elements into IDEs that intrinsically motivate developers and end-user programmers. Since open source software (OSS) features collaboration, and methods to make or suggest improvements, OSS model could serve as a referential starting point.

A common assumption in modern society is that developers are motivated by external factors such as financial incentives, career success or peer approval. However many studies demonstrate that extrinsically motivating a person increases their ability to perform non-complex tasks [26,18]. Once a task becomes complex and requires creativity and problem solving, extrinsic motivating factors have been shown to decrease performance [26,18].

Since extrinsically motivating developers could undermine their intrinsic motivation [15] and the competencies exhibited by superior programmers, strategies to increase intrinsic motivation become increasingly important. In fact several studies suggest developers rarely begin coding for financial gain, [7] with less than 20% report becoming developers for money. The remainders are attracted by the development process itself and would not switch careers for a significant increase in salary [25].

Since most developers rely on some sort of IDE, and little work has been focused on motivation in this area [7], we examined agents to promote intrinsic

motivation in that environment. There is also a lack of an appropriate method to measure developer motivation as previous work has typically used the unreliable metric of job turnover, which encompasses lifestyle and family pressures [7]. Under that premise, we explore the meta-theory of self-determined behavior as a suitable lens to view motivation in software engineering.

Self-deterministic theory (SDT) posits that an individual feels the most intrinsically motivated when they are experiencing a high level of the three basic psychological needs of autonomy, competence and relatedness [19,42]. Henceforth, this paper will refer to these components as the three basic needs. An individual experiences autonomy when they have self-directing freedom [19]. In an IDE, a user would experience autonomy when they can choose to work in the manner they wish. If an IDE forces certain work actions that a user doesn't willfully choose they may experience decreased autonomy.

Competence is a sense of capableness coupled with the confidence that a person can achieve an imminent task [19]. This could translate to the IDE environment in the form of learnability, usability and debugging. If an IDE has too steep of a learning curve, is an environment that causes developers to error easily or cannot effectively debug, the developer may feel a reduced sense of competence. A reduction in the perception of competence could also occur if attaining assistance or necessary technical information is too difficult.

Relatedness encompasses the engagement a developer has with his or her direct peers and technical community [19]. An IDE that supports relatedness would allow for collaboration, make communication between teams efficient and provide means to develop trust in a teammates work.

Creating an IDE that sufficiently supports and avoids undermining feelings of autonomy, relatedness and competence could make large strides in increasing innovation and diversity in the computer science community.

Previous studies have evaluated intrinsic motivation in virtual environments [11,44], however none have measured the level of intrinsic motivation experienced while using an IDE. In response we created a validated survey tool measuring intrinsic motivation while engaging with a software tool.

While there is a large body of knowledge studying motivation in software developers, there is little through the lens of SDT. We listed the methods of supporting and undermining motivation in the computer science literature and compared it to the techniques in the psychology knowledge base. Using the survey tool and the literature analyses, we seek to answer the following questions:

1. Do modern IDEs support intrinsic motivation in developers?
2. What types of developers experience the most and least intrinsic motivation?
3. How can modern IDEs be adapted to foster heightened and reduce undermined intrinsic motivation?

In addition to these research questions we completed a literature summary of the computer science and psychology fields to assess the relevance of an SDT model to view intrinsic motivation in developers.

4. Self deterministic theory explained

Since we have chosen to use SDT as a model to assess a developer's feelings of motivation while using an IDE, it is important to understand its constructs and how they relate to each other. The following sections explain SDT and offer insight into behaviors that support and undermine intrinsic motivation. The constructs are interdependent and their relationships are shown in Figure 1.

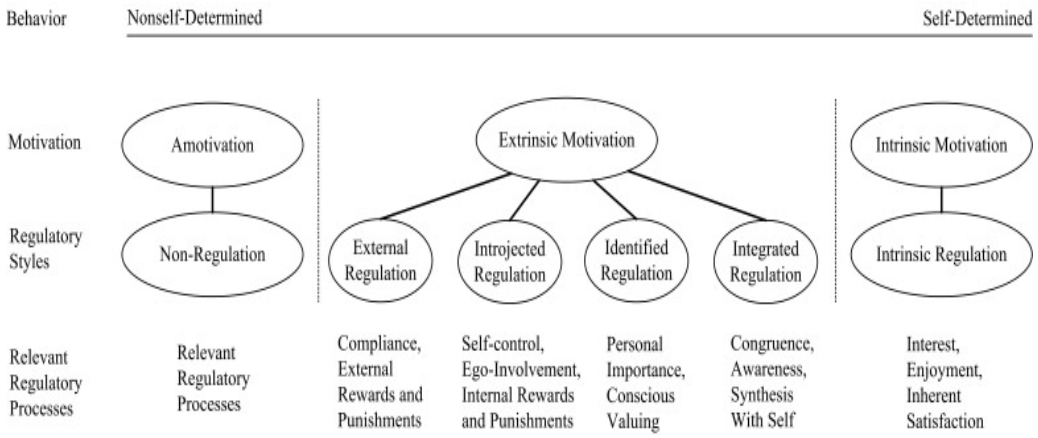


Figure 1: The self-determination continuum [42]

4.1 Self-determined behavior

The primary difference between and intrinsically and extrinsically motivated behavior is source of causality. If the reason to behave in a certain manner is decided by the individual performing it, it would be described as self-determined. In relation to SDT, self –determined behavior exists on a continuum, see Figure 1. Since the most self-determined behavior is intrinsically motivated it stands to reason that it enhances the traits of exceptional developers. It is defined as the attitudes and abilities needed to act as the primary locus of control in a person's life. The most self-determined behavior exists when people make

choices about their quality of life without unnecessary external influence or interference [57]. In short, being self-determined means being the causal agent in the choices and decisions impacting one's life [19,20,58]. Self-determination is influenced by the numerous factors detailed in the sections below.

4.2 Self-Deterministic Definitions

Intrinsic, extrinsic motivation and amotivation are the drives experienced by an individual directly impacting their ability to act in a self-determined manner [42].

Intrinsic motivation is the drive to engage in an activity for its inherent satisfactions while striving inwardly to be competent and self-determined [42]. It can be observed in infancy as exploratory behavior, is mutually exclusive of parental encouragement [9] and driven by the basic psychological needs of autonomy, relatedness and competence [42]. Even though intrinsic motivation is an innate trait, the level at which it’s experienced depends greatly on social context. As a result, it is easily disrupted by unsupportive conditions [43]. When an individual feels intrinsically motivated there are numerous benefits. They experience an elevated sense of well-being, a greater desire to engage in challenges, heightened persistence, creativity and conceptual understanding [16,41,42,49].

Extrinsic motivation is the drive to engage in an activity based on external rewards or constraints imposed by others. Like intrinsically motivated actions, extrinsically motivated behaviors can be self-determined if an individual internalizes it [42]. This process is organized across a continuum based on perceived loss of causality and is organized into four categories of self-regulation; external regulation, introjected regulation, identified regulation and integrated regulation [42]. External regulation is typically experienced as being controlled or alienated [14]. It is the type of motivation studied by Skinner [51] and the

common association of the general population to the concept of motivation. Introjected regulation involves absorbing a behavior but not fully accepting it as one's own. These behaviors are contingent on self-esteem and are often performed to avoid guilt and anxiety and can support ego in the form of pride. A more autonomous regulation is that of identification, which happens when a person consciously values a behavior and accepts it as personally important [42]. The most self-determined type of extrinsic motivation is identified regulation. A behavior is identified when it has been evaluated and synthesized with the individuals existing values. While these actions share qualities associated with intrinsic motivation, they are completed to attain an external outcome and thus classified as extrinsic [42]. Additionally attaining an external reward diminishes the experience of intrinsic motivation for the activity performed [42].

Amotivation is the state of lacking the intention to act. [19]. It is similar to the concept of learned helplessness [2] that occurs when individuals feel incompetent, repressed volition and do not see any connections between their actions and an outcome [19]. Amotivation results from placing no value in an activity [19] not feeling the competence to do it [4], or expecting it to yield an undesirable or nonexistent outcome [47].

4.3 Self- deterministic theory

Self-deterministic theory (SDT) is a meta-theory encompassing a broad framework of synthesized research and theories demonstrating how to best elicit self-determined behavior. It encompasses inherent growth tendencies and the innate psychological needs of autonomy [14,17] relatedness [5,45] and competence [29,38] as a basis for self-motivation and personality integration. The basic needs are vital to the experience of intrinsic motivation, integrity and well-being and must be sustained across the life span [37,55]. Each need can be enhanced or diminished in different social contexts, detailed below.

Autonomy is the need to experience self-direction and personal choice in the engagement of one's behavior. Behavior is autonomous when one's interests, preferences and wants guide our decision-making. There are three subjective qualities within the experience of autonomy: an internal perceived locus of causality, volition, and perceived choice [19,42]. It is feelings of autonomy that directly impact the previously described self-regulation categories of extrinsic motivation. When a person makes a choice based in personal endorsement rather than compliance, more autonomy is experienced and integrated regulation is most likely to occur [40,54]. Finally, autonomy has often been perceived as detrimental to relatedness and community due to its association with individualism and independence [52], which would seem counterintuitive to SDT. Within SDT however, autonomy refers to the feelings of volition rather than independence that accompany individualistic or collectivist acts [42]. This finding is supported by a study [31] conducted in Korea and the U.S., which discovered a positive relationship between autonomy and attitudes of collectivism rather than between autonomy and individualism.

Relatedness is the need to experience a sense of belonging, inclusion, close emotional bonds and attachments into a specific group of individuals. Like autonomy, relatedness is needed on order to facilitate internalization of an extrinsically motivated behavior [42]. Persons in relationships with elevated relatedness understand each others rationale for prescriptions and proscriptions. These relationships are characterized by emotionally positive interactions and partners, as well as perceptions of an intimate, high quality, caring, liking, accepting and valuing social bonds. [42]. They are also associated with belief in surplus, lack of judgment, shared experience, self compassion, introspection, low entitlement, high agreeableness and extraversion. Unsupported relationships are associated with a belief in scarcity, low self-compassion, low introspection, anxiety and high entitlement [13]. Finally, higher related feelings occurred in

exchange relationships, where there is a trade of favor with some future expected return [10] when there was a high level of trust [30] and an obligation to reciprocate [10,27].

Competence is the need to be effective in interactions with the environment. Those experiencing feelings of competence perceive that they can excel at the task at hand [42]. Persons also feel elevated competence when they experience flow; a state of concentration that involves a holistic absorption in an activity [27]. Certain aspects of flow are regulated by task difficulty and individual skill set, meaning if the task is too difficult or easy for an individual, flow cannot be experienced [27]. Flow is also supported through the concept of optimal challenge where a person experiences the most pleasure following success with a moderately leveled challenge [29].

4.3.1 Self-deterministic sub-theories

Five sub-theories form the basis of the self-deterministic meta-theory. These theories were formed by studying certain behaviors and approaches impacting intrinsic motivation and thus become important in determining appropriate methods to use in the computer science field.

Cognitive Evaluation Theory (CET) was framed by Deci and Ryan [19] in terms of social context and environmental factors that facilitate versus undermine intrinsic motivation. It primarily focuses on the needs for autonomy and competence and shows that elevated intrinsic motivation cannot be accomplished unless feelings of competence are accompanied by a sense of autonomy [22,39].

Organismic Integration Theory (OIT) details the factors that promote or suppress internalization and integration of external motivators. Internalization is more likely to occur when there is environmental support for relatedness. Feelings of volition elicit internalization and are a critical element for integration [19,21,42].

Causality Orientation Theory (COT) describes and assesses three types of behavior regulations; the autonomy, control and impersonal orientations. The autonomy orientation deals with action based in interest and value for a situation. Control orientation focuses on rewards and gains, while impersonal is characterized by anxiety concerning competence [42].

Basic Psychological Needs Theory (BPNT) suggests psychological well-being is predicated by experiencing competence, relatedness and autonomy. All three needs are essential and universal aspects of wellness across cultures. If any of the three needs are not experienced it results in reduced well-being [21].

Goal Contents Theory (GCT) highlights the differences in intrinsic and extrinsic goals and maps their impact on motivation and wellness. Extrinsic rewards are aligned differently than intrinsic and have a negative impact on well-being [21].

5. Related Works

While there has been significant work done in the area of motivation in developers, nothing has been completed specific to SDT in an IDE. Much of the research supports SDT, but a significant portion was completed before the 2000 publication introducing the theory [42]. SDT and its direct constructs has been used to examine end-users in the virtual environments of gaming, online learning and open source, but none have looked at developers.

5.1 SDT and Computer Use

SDT in virtual environments and video games

Chen and Jang [11] drew on SDT to measure online learner motivation and concluded that motivation and self-determination did not predict learning outcomes. The study did however, support SDT's main theory of distinctive motivational constructs.

Another study investigated motivation for computer based game play and its effect on well-being [44]. They concluded that features conducive to perceptions of autonomy, competence and relatedness enhance motivation for game play, leading to enhanced well-being. Autonomy and competence elicited feelings of intuitive controls (sense of control and effectiveness), game enjoyment and desire for future play. Competence was also related to presence, higher state of self-esteem and mood.

The motivation for end-users to appropriate open source software (OSS) was examined using the academic motivation scale and the three types of motivation encapsulated in SDT: intrinsic, extrinsic and amotivation [32]. Results indicate that with the exception of the need to accomplish, all other motivational factors were relevant.

Motivation in Software engineering

Motivation in software engineering has been studied in many capacities outside of SDT but little work has been done to examine the known developer motivators of learning, exploring new techniques and problem solving [7]. A literature review by Beecham et. al developed a new software engineering motivational model, Motivators, Outcomes, Characteristics and Context (MMOC) based on those 4 characteristics identified in previous literature [48]. The model proffers a method for software engineers and managers to gain insight into their behavior and includes newer works focusing on open source where facets such as turnover and productivity are minor. This study also surveyed motivation in software engineering and produced several themes.

Individual tendencies such as moderating paired with controlling or implementing play a role in whether software engineers form homogenous groups.

They found 22 different motivators and 15 de-motivators for software engineers, some of which fell into both categories. This was thought to be due to varying career stages of the individuals studied. Turnover and absenteeism are the most common outcome of engineering de-motivation, while little work focused on motivations to stay in the profession.

While learning, exploring and problem solving are motivating aspects of software engineering, scant work has examined their nature or how the reliance on tools or programming languages are impacted by them. Better ways of measuring motivation need to be developed.

There is a clear need for a comprehensive model of motivation in software engineering that includes the motivating factors in the task of software engineering itself. There are a variety of software engineering models, none

address the characteristics of moderating and controlling, moderating and implementing [7].

It is important to note that SDT is not one of the theories addressed in the Beecham et. al. study, however most of the sub theories and concepts attributing to it are.

A second literature review completed in 2011 [23] again summarizes motivation in software engineering. They concluded that even though the number of researches in this area has increased in recent years the overall understanding of what actually motivates software engineers hasn't changed significantly.

Finally the most recent study in this area has been a qualitative analysis involving a case study of a small software engineering company [24]. They concluded that learning and growth contributed the most to the motivation story at the company. Like the majority of existing software engineering research, it supports SDT as an appropriate model to evaluate motivation.

OSS and motivation

Because of its unique characteristics it is important to understand the motivation of OSS developers. The OSS environment is different from traditional development in that it is always a collaborative effort using different communication styles. OSS developers can make contributions to the software they use and there are stringent systems for determining competence of these code committers. These differences are directly related to the basic psychological needs so one could anticipate that OSS developers would have a different motivation score than traditional developers. It is important to measure the motivation of these developers because they use and also create IDEs.

Several studies examined developers' motivations to be involved in OSS. Bitzer et. al. [8] analyzed the phenomenon of OSS, and motivation for sharing from an economic stand point. They showed that the intrinsic motivating factors

of fun, play, and gift culture are motivators in OSS involvement. Since many OSS projects are successful and profitable, it negates the idea that intrinsic motivators shouldn't be considered in commercial software for economic gain.

Linux contributors were surveyed by Hars et. al [28] to determine if their motivation factors were intrinsic (innate desire to code, and altruism), or extrinsic (future revenues, building human capital, self marketing, peer recognition, personal needs). Their results indicated that OSS developers had a higher propensity for extrinsic factors heavily weighted in building human capital. This uncovers a design flaw in thier study in that building human capital is commonly perceived as expanding a skill set. Furthermore it is explicitly defined as such in the Hars paper. Viewed this way, building human capital is a learning and exploratory behavior which would be classified as an intrinsically motivated behavior.

End-user programming and motivation

Because of their potential impact to the future innovation in ubiquitous computing, it is meaningful to examine the literature surrounding end-users. In addition to the existing motivational research in computer science, two studies deal with end-user programming and motivation. They demonstrate that SDT constructs are well supported in terms of end-user programmers and that the survey tool created in this paper would effectively measure the motivation of end-users relying on tools to develop. Chintakovid [12] researched flow and self-efficacy, which are correlated to the basic need of competence. In this case the more self-efficacy an end-user programmer has the more flow they experienced. Beckwith et. al [6] examines the effect of self-efficacy and tinkering (to encourage self efficacy) on performance in end-user debugging with respect to gender. Females who tinkered felt more self-efficacy and in turn increased performance.

6. Methodology

6.1 Vetting SDT as an appropriate model for analyzing motivation in computer science.

To evaluate how suitable SDT is to measure motivation in computer science, we turned to the existing literature. We summarized motivators and de-motivators from the computer science research and methods effecting intrinsic motivation in psychology literature. These summaries were then synthesized into specific recommendations for improving intrinsic motivation in IDEs.

Previous work by Beecham et. al [7] provided us with lists of motivators and de-motivators in the existing literature and the number of citations supporting them. 519 papers were surveyed and 22 motivator and 15 de-motivator themes were identified. This list provided the basis for our analysis of SDT as an appropriate theory of motivation for developers. Each motivator and de-motivator were vetted and categorized into the constructs of SDT; autonomy, relatedness, competence, and intrinsic and extrinsic motivation. Additionally, they were labeled as supporting or undermining the three psychological needs. If a motivator or de-motivator couldn't be categorized, it was classified as 'unclear'. Finally each motivator and de-motivator was assigned an identification tag to ensure its source when they were synthesized into final recommendations to improve IDEs. To determine if the constructs of SDT were indeed relevant to computer science, we totaled the number of citations supporting each. See Tables 1 and 2 in the Appendix for a detailed depiction.

To determine the common methods for enhancing and undermining intrinsic motivation in the psychology literature, we surveyed 23 papers representing the most relevant work in SDT. [2,5,9,13,14,16,18,19,20,21,27,29,30,37,38,41,42,43,45,49,52,54,55,57,58]. Our

analysis identified 30 thematic methods of supporting and 25 undermining the three basic psychological needs of autonomy, competence and relatedness. All methods were indentified as intrinsic motivators tagged with its relevant basic need. Autonomy was supported in 12 methods and undermined in 7, competence was supported in 7 methods and undermined in 7, and relatedness was supported in 12 and undermined in 7. Each method was assigned an identification number so that we could synthesize them with results of the computer science literature review before making our recommendations to improve IDEs. See Appendix Tables 3, 4 and 5 for details.

Next, to determine if SDT was a suitable theory of motivation in computer science, we tallied the number of citations supporting each SDT construct, see figure 2. A total of 469 citations indicated at least one of the SDT constructs was a factor in determining developer motivation. Intrinsic factors supporting motivation were referenced 167 times while undermining surfaced 30 times. The basic needs were also indicated at a high rate and 12 citations were excluded because we couldn't determine a supporting construct.

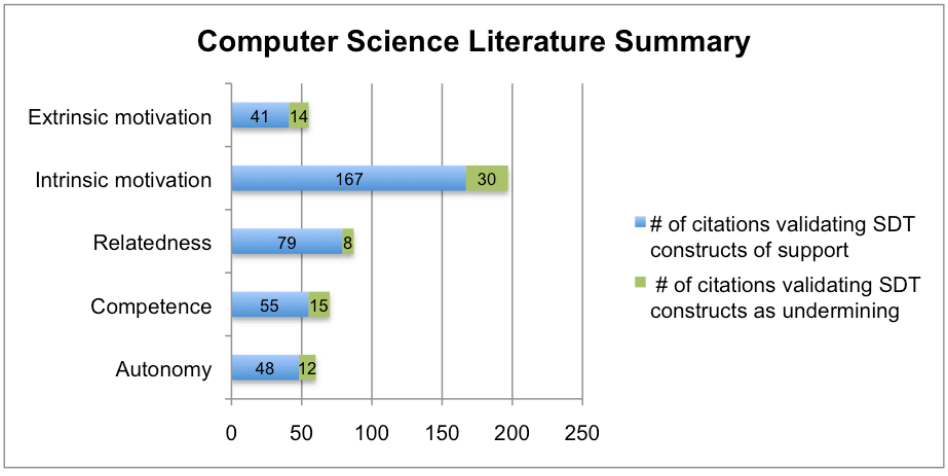


Figure 2: Summary of literature review citations related to SDT

Finally, we synthesized the two literature analyses into approaches for improving intrinsic motivation in IDEs. To ensure the specific approaches we prescribed were rooted in existing research, we combined both the psychology SDT methods and the computer science, see appendix table 6. Based on this list, we hypothesized 16 specific approaches to increasing intrinsic motivation in an IDE. Each approach was tagged with its relevant basic needs, supporting psychology methods and motivators/de-motivators from the computer science literature. This was done to visualize which approaches could have the most potential impact on enhancing intrinsic motivation in an IDE. See Table 6 in the Appendix for details.

6.2 Validated tool to measure the basic need support

Part of the problem in studying developer motivation is the lack of a consistent method to evaluate motivation. Additionally there is currently no method to evaluate intrinsic motivation support while using a software tool. Since higher feelings of autonomy, relatedness and capableness have been shown to support feelings of intrinsic motivation we developed and validated an instrument to measure basic need support while using a software tool (see Table 6.1). To do so, we surveyed developers in a diverse demographic who answered 5 questions each for each basic need measuring how autonomous, related and capable they felt while using their development tools see table. The questions used were generated based on existing basic need support scales [29,59,60,61,62] and adapted to IDE use. The tool was validated with 103 responses for reliability in each basic need section; Cronbach's alpha autonomy (.815), relatedness (.927) competence (.815). Validity factor analysis showed a one-factor solution for each need. Based on the average of the 5 corresponding questions each participant received a score indicating how autonomous, capable and related they felt while

using their current tool set. They also received a total intrinsic motivation support score encompassing the average of all three needs. The terms defined in the survey were, *My tools,* the collection of tools that you use to program, design or create user experience, *Autonomy,* the freedom to function or act independently and un-coerced (This refers to your experience within the tool rather than your experiences with colleagues), *Relatedness,* building effective relationships in your team while fostering interpersonal support, and *Competence*, the ability to do something successfully or efficiently.

Table 6.1 Validated survey tool to measuring intrinsic motivation support

Autonomy
How strongly do you agree with the following statements describing your sense of autonomy while using programming, software design and architecture and user experience design tools? Scale 1-7 (1 = strongly disagree, 7 = strongly agree)
My tools allow me to customize my experience.
My tools allow me to present my work in a way that others can understand.
My tools allow me work in the manner I wish.
My tools allow me to find the answers that I need.
My tools allow me to voice concerns or suggestions for improvement.
Relatedness
How strongly do you agree with the following statements describing your sense of relatedness while using programming, software design and architecture and user experience design tools? Scale 1-7
My tools have features that facilitate team member connectedness.
My tools allow me to understand how my team mates work.
My tools provide me with the necessary information to build trust in the work of my teammates.
My tools allow me to collaborate with my teammates.
My tools make it easy to communicate with my team.

Competence
How strongly do you agree with the following statements describing your sense of competence while using programming, software design and architecture and user experience design tools? Scale 1-7
I am skilled at using my tools in comparison to other developers or designers.
I felt competent after working with my tools for a month.
I am proficient at using my tools.
I am satisfied with my performance while using my tools.
I feel able to meet project challenges using my tools.

6.3 Assessing intrinsic motivation in developers

We questioned 103 developers and assessed their intrinsic motivation, demographics, development methods and preferences. These developers were recruited via online outlets, e-mail, social media and word of mouth. Persons under 18 years of age were excluded as well as developers with less than 3 months of experience. We hoped to identify patterns in the developers work style that would identify specific areas where they felt lacking in intrinsic motivation. Each developer was surveyed about their years of experience and percentage of time they spend programming (writing, debugging, and maintaining the source code of computer programs), software architecture and design (the process of problem solving and planning for a software solution including low-level component and data structure design, the architectural view and identification of technical and user requirements), and user experience design (the design of all aspects of a users experience with the system including the interface, graphics, physical interaction, and the manual). We also assessed intrinsic motivation based on developer style and preference. They rated on a scale from 1-7 feelings about

version control, WYSIWYG (what you see is what you get) graphical user interface tools, code reuse, testing and peer programming.

Respondents were also questioned about their intrinsic motivation with regard to a large number of software development tools. The tools ranged in purpose from programming to architecture to user experience and the respondents were asked to indicate which tools and languages they favored. If the developer didn't use the tool listed, no data was recorded for that tool. We included the following list of tools and languages. Tools: Adobe Creative Suite, Adobe Flash Builder, AppMobi, Aptana, Axure, Basalmiq, Browser developer tools, Cloud 9, Eclipse, Emacs, Invision, Iplotz, iRise, JetBrains, IntelliJ, IDEA, JetBrains, WebStorm, jQuery, Justinmind, Microsoft Expression Blend Studio, Microsoft Web Matrix, MobiOne, manual sketches, Phonegap, Power Point, Protoshare, Sencha, Architect, VIM and Visual Studio XAML/WPF. Languages: ActionScript, Apex, ASP, C/C++, C#, Clojure, CoffeeScript, ColdFusion, Erlang, F#, HTML/CSS, Java, JavaScript, Objective-C, Perl, PHP, Python, Ruby, Scala, Shell Script, SQL, Visual Basic and XAML.

7. Results

7.1 Do contemporary IDEs support motivation in developers?

Developer demographic

103 developers with diverse backgrounds responded to our survey. The average age was 34 years and ranged from 18-65 years of age. Males accounted for 53% of survey respondents, and females 47%. While 97% of them currently reside in the US, they originated from 16 different countries. The developers ranged in experience from 0.25 yrs to 33 yrs, with an average of 8 yrs of experience. 71% have developed as a hobby while 86% have coded as a means of primary income.

Our respondents engage in multiple types of development. Mobile application development was practiced by 46%, web development 79%, desktop 63%, databases 61%, embedded 19% and system development 46%. As to job function, they spent an average of 45% of their time programming, 22% engaged in activities related to software design and architecture and 18% on GUI design and user experience. They came from small companies of 1 employee to over 5000, with an average of between 51 and 200 employees. Students (undergrad and graduate) made up 48% of the respondents while 52% were not. Their academic backgrounds were primarily computer science (55%), natural sciences (14%), math and statistics (12%) and engineering (12%).

Developers and tools

Our survey found that developers have an average intrinsic motivation support score of 4.85 on a scale of 1-7 while using their tool sets. A score of 7 indicates strong feelings of intrinsic motivation. This potentially shows that there is room for improvement, that tools could be developed to better support intrinsic motivation, autonomy, competence, and relatedness.

Most of the tools surveyed did not have a significant relationship with intrinsic motivation in any capacity. Some demonstrated correlations to competence, but no IDE showed a significant relationship to relatedness or autonomy emphasizing the lack of community and flexibility in the developer's experience. Developers that used Adobe Creative Suite and Visual Studio felt the highest competence ($r = 2.51$, $p < 0.05$). Finally, there were no significant positive correlations with autonomy or relatedness in any IDE, suggesting a need for more flexible, connected tools.

Two areas we hypothesized would have a significant relationship, didn't. There was no significant relationship between intrinsic motivation support and IDEs typically used in developing Open Source Software, such as Eclipse. There was also no significant relationship between intrinsic motivation and respondents using prototyping tools. Table 2 provides an overview.

Table 7.1 Spearman's Correlation Matrix r Tools vs. basic need scores * indicates **p<.05** ** indicates **p<.01**

	Intrinsic Motivation Support Score	**Competence Score**	**Relatedness Score**	**Autonomy Score**
Adobe Creative Suite is favorite tool	-	.285**	-	-
Visual Studio, XAML/WPF as part of my job	-	.230*	-	-
Visual Studio, XAML/WPF is favorite tool	-	.324**	-	-

7.2 What factors affect intrinsic motivation support in developers?

Coding in spare time

A significant relationship was found between the 15% of developers who contribute to OSS projects in their spare time and the intrinsic motivation support score ($r = .220$, $p < .05$). While there was no significant correlation with competence or relatedness, there was with autonomy ($r = .216$, $p < .05$).

Developers that spent the most time exploring and learning new development technologies in their spare time had a positive correlation with intrinsic motivation support ($r = .222$, $p < .05$) and autonomy ($r = 264$, $p < .05$).

Those who developed in their spare time for a paying customer had a significant positive relationship with competence ($r = .324$, $p < .01$).

Developer roles

Developers whose time was spent primarily programming rather than in software design or user experience design showed a significant positive correlation with autonomy ($r = .336$ p, < 0.01). No significant correlations between intrinsic motivation and other activities related to programming surfaced.

Code Re-use

49% of the respondents felt that it was easy to determine the quality of code that they re-use. These developers also had positive correlations to intrinsic motivation ($r = .305$, $p < .01$) and relatedness ($r = .265$, $p < .05$).

Version Control

According to 73% of developers surveyed, version control is well integrated into their programming process. These respondents had a positive correlation with intrinsic motivation support ($r = .227$, $p < .05$), competence ($r = .259$, $p < .05$) and autonomy ($r = .252$, $p < .05$). On the other hand 20% of the

respondents found it difficult to track what others were doing with code in repositories. These developers had a significant negative correlation with all for measures. Intrinsic motivation correlation was ($r = -.446$, $p < .01$), competence ($r = -.261$, $p < .05$), relatedness was ($r = -.359$, $p < .01$) and autonomy was ($r = -.316$, $p < .01$).

WYSIWYGs

Developers who use WYSIWYGs to design GUIs had a significantly negative correlation with intrinsic motivation support ($r = -.296$, $p < .01$), relatedness ($r = -.231$, $p < .05$) and autonomy ($r = -.240$, $p < .05$)

Gender

Female developers showed a significantly negative correlation with feelings of competence ($r = .242$, $p < .05$)

Finally, we did not find a significant correlation for between the 39% of developers who use peer programming and relatedness. Developers with the most experience and a background in computer science did not significantly correlate with competence. Also, the developers who engaged in the most different types (mobile, web, embedded etc.) of development did not exhibit a significant relationship to competence.

Table 7.2 Spearman's Correlation r matrix; basic need support score vs. developer characteristics *indicates $p < 0.05$, **indicates $p < 0.01$

Developer Characteristic	Intrinsic Motivation Support Score	Competence Score	Relatedness score	Autonomy Score
Explore or learn new development technologies as a hobby	.222*	-	-	.264*
Develop projects for a paying customer outside of day job	-	.324**	-	-
Contribute to open source projects as a hobby	.220*	-	-	.216*
High percentage of time spent programming (rather than architecture or UX design)	-	-	-	.336**
Find it easy to determine the quality of code they reuse.	.305**	-	.265*	-
Feel version control is well integrated into my programming process.	.227*	.259*	-	.252*
Find it difficult to track what others are doing with the code in the repository.	-.446**	-.261*	-.359**	-.316**
Use WYSIWYG tools when developing a GUI.	-.296**	-	-.231*	-.240*
Females	-	-.242*	-	-

7.3 How can modern IDEs adapt to foster heightened intrinsic motivation?

Approaches to enhance intrinsic motivation in an IDE

Our literature audit resulted in 13 approaches to enhancing intrinsic motivation in an IDE, see table 4. Here, we compare these approaches to our survey results to better visualize their potential impact. We assigned each approach the number of methods from the literature supporting it. We also indicated whether it was supported by our survey and the three basic psychological needs of autonomy, relatedness and competence. We determined any approach that facilitated feelings of autonomy, competence or relatedness to be supported by our survey results. Competence was selected because only 2 out of the 31 tools listed significantly correlated positively with competence. 3 tools also had a significant negative relationship with competence. Autonomy was chosen because 4 of the listed tools significantly correlated negatively with autonomy. Additionally, there was no significant positive relationship between autonomy and the remaining tools. Finally we selected relatedness because it did not have a significant relationship with any tool.

The most strongly supported approach was #1: Create a moderated environment between users fostering positive social interactions and expression of preferences, interests and competencies. The environment would also need to suppress feelings of entitlement and encouraging feelings of equity. This approach was supported by 11 methods in the literature, our survey results and all the three basic psychological needs of autonomy, relatedness and competence.

Approach #2: Supply avenues for developers to express concerns and work style preference to the IDE creators. Paraphrase back concerns and ask developers for input, was supported by 8 methods in the literature, all three basic needs and our survey results.

Approach #3: Complete usability studies on features to ensure clear communication, reduce any unnecessary steps and test intrinsic motivation score. Testing should also ensure structured workflow; make sure it's in optimal challenge, is also well supported. It is supported by all three basic psychological needs, our survey results and by 6 methods in the literature. This approach also employs the tool we created to measure intrinsic motivation. The average intrinsic motivation score in our survey was 4.85, using this tool in usability testing could quantify feelings of intrinsic motivation in future IDEs

Finally approach #4: Provide a means for developers to collaborate in solving problems and supply easy channels to find information about their problem, is support by all three basic needs, our survey results and 4 methods from the literature. This approach would need to look beyond simple peer programming, which wasn't indicated as particularly intrinsically motivating in our survey.

There are an additional 9 approaches that are less supported, but with reasonable potential to improve intrinsic motivation in IDEs shown in table 4.

Table 7.3 Approaches to support intrinsic motivation in an IDE

Autonomy	**Relatedness**	**Competence**	**Supported in survey results**	**# of supporting methods in the literature**	**Synthesized approaches to enhance intrinsic motivation in an IDE [2,5,9,13,14,16,18,19,20,21, 27,29,30,37,38,41,42,43,45,49,52,54,55,57,58]**
✓	✓	✓	✓	11	1. Create a moderated environment between users fostering positive social interactions and expression of preferences, interests and competencies. The environment would also need to suppress feelings of entitlement and encouraging feelings of equity.
✓	✓	✓	✓	8	2. Supply avenues for developers to express concerns and work style preference to the IDE creators. Paraphrase back concern and ask developer for input.

✓	✓	✓	✓	6	3. Complete usability studies on features to ensure clear communication, reduce any unnecessary steps and test intrinsic motivation score. Testing should also ensure structured workflow; make sure it's in optimal challenge.
✓	✓	✓	✓	4	4. Provide a means for developers to collaborate in solving problems and supply easy channels to find information about their problem.
✓	✓	✓	✓	1	5. If certain methods or work styles are forced, have clear rationale available.
-	✓	✓	✓	5	6. Identify lacking developer skills and coach them to improve while communicating the importance of failing and its relation to future success.
-	✓	✓	✓	3	7. Introduce features that assist developers in understanding how their teammates code and work. Additionally, add features that facilitate a quicker return to state of flow via methods that aid developers back into the code they produced earlier or by someone else.
✓	-	✓	✓	6	8. Use performance metrics to identify developer amotivation, determine lacking psychological need and remind programmer of relevant supporting features.
✓	-	-	✓	5	9. Ensure developers have the freedom to choose coding style and preferences.
-	✓	✓	✓	2	10. Add humorous components and positive feedback when the developer succeeds.
✓	✓	-	✓	2	11. Mimic the sense of community in open source environments and encourage some level of initiative and empowerment in tasks.
✓	-	✓	✓	2	12. Endorse skill sets and provide certifications pertaining to career growth.
✓	-	-	✓	2	13. Ensure developers can work from multiple locations with appropriate access if online.

8. Discussion

In this section we discuss the feasibility of using SDT in relation to developer motivation and the outcome of doing so.

Is SDT a good model for measuring motivation in developers?

Our findings demonstrate that SDT is an appropriate model to measure levels of intrinsic motivation in developers. 98% of the citations in existing literature support the one or more constructs of SDT defined by [42] in 2000. We also determined that our survey tool is robust method of determining intrinsic motivation levels while an individual is using a virtual tool.

Do modern IDEs support intrinsic motivation in developers?

Our survey indicated that there is significant room for improvement in the intrinsic motivation developers feel while using their tools. On average developers had a total intrinsic motivation support score of 4.85 out of 7. Each basic psychological need score returned a similar result indicating the need for at least 20% improvement in the areas of autonomy, competence and relatedness while using an IDE. This indicates that while developers aren't feeling a deficit of intrinsic motivation, they aren't feeling a high level either. If future IDEs created an environment rich in feelings of intrinsic motivation, a more diverse group of developers might be compelled to code. This could ultimately lead to increased innovation and disruptive technologies.

Visual studio and Adobe Creative Suite were the only tools that had a positive relationship to competence. The other 29 tools showed no significant positive relationship to the basic needs highlighting a large gap in intrinsic motivation and tools that facilitate programming, software design and UX design. It appears that IDEs aren't making developers feel very competent. The more developers that feel competent, the more likely they are to even attempt a project.

Raising those numbers could greatly enhance innovation. It is important to note that the lack of positive or negative correlation could have been because the developers in our survey didn't use them. We don't believe this to be the case, because all but 3 tools were used by more than 20% of the developers surveyed.

Since Eclipse was an IDE developed as an open source tool, we would have expected developers who use it to have more feelings of autonomy, competence and relatedness. Surprisingly our survey results did not indicate a significant relationship between intrinsic motivation and the use of Eclipse. It may be that the developers in our survey using Eclipse weren't part of the OSS community contributing to it, thus weren't reaping the benefits.

It is also important to note that we did not find a significant relationship between for any intrinsic motivation constructs in respondents using prototyping tools. We originally hypothesized that these individuals would feel less related because of the current disconnect between IDEs and prototyping.

What types of developers experience the most and least intrinsic motivation?

Developers were asked about the reasons they code outside of work and several correlated positively to intrinsic motivation support. We did not ask about work related reasons therefore we couldn't look for a relationship there. Those who contribute to open source projects in their spare time showed a positive correlation to intrinsic motivation. This is not surprising since those involved in open source projects also engage in related communities and have the autonomy to dictate solutions to the problems they encounter. Because of this, OSS projects can serve as inspiration to improve intrinsic motivation.

Developers that spent the most time exploring and learning new development technologies in their spare time correlated positively with intrinsic

motivation support and autonomy. This is not surprising as intrinsically motivated people are driven to explore and learn.

Those who developed in their spare time for a paying customer correlated positively with competence, which also makes sense. If a developer is confident that someone would pay for their moonlighting work, they are likely to feel competent in their skill set.

Intrinsic motivation support was high in developers whose time is spent primarily programming rather than in software design or user experience design. No significant correlations between intrinsic motivation and other activities related to programming surfaced, suggesting a need to increase competence and relatedness in features associated with basic programming. Since there were no significant correlations between intrinsic motivation and other activities related to programming, it seems that features supporting software design, architecture, GUI design and UX are severely lacking in intrinsic motivation support overall.

Developers who felt it was easy to determine the quality of the code they re-use felt more intrinsic motivation support and relatedness. This seems likely considering the implications of relatedness and understanding someone else's work [42].

Version control was another area showing positive relationships to intrinsic motivation. If developers felt that version control was well integrated into their programming process higher feelings of competence and autonomy. Conversely the developers finding it difficult to track what others were doing with code in repositories had lower feelings of intrinsic motivation, autonomy, relatedness and competence. This confirms our hypothesis and Deci's [42] work that being able to understand the work of your peers is imperative to your feelings of relatedness.

Finally, developers who used WYSIWYGs to build GUIs showed a negative correlation to intrinsic motivation, autonomy and relatedness.

WYSIWYGs don't allow developers as much control in their GUI code which would explain reduced feelings of autonomy.

Strengthening methods to understand developer actions in code repositories may be a great place to start in improving feelings of autonomy, competence and particularly relatedness.

With regard to developer demographics, the only significant correlation was negative. Female developers had low feelings of competence. This is to be expected as previous work has shown women developers feel less self-efficacy than their male counterparts [6].

We hypothesized that developers engaging pair programming would feel higher relatedness, however that was not the case in our study. It could have been due to low numbers, but 39% of our respondents said they currently pair program. It may be a result of the potentially negative feelings developers could experience when someone is looking over their shoulder pointing out errors.

We would have also expected that developers with the most experience or a background in computer science would have higher feelings of competence than those who didn't. However, there was no significant relationship present in any of these cases. Finally, we expected higher feelings of competence in the developers who engage in multiple types (mobile, web, embedded etc.) of development. Our study did not reflect this, which may indicate that it's not an existing skill set, but rather factors such as self-efficacy that determine feelings of competence.

How can modern IDEs be adapted to foster heightened and reduce undermined intrinsic motivation?

Based on our literature review and our survey, four approaches surfaced as having the most potential impact in enhancing intrinsic motivation in future IDEs.

1. Create an environment moderated for positive interaction that allows developers to interact socially and express preferences, interests and

competencies. The environment would also need to suppress feelings of entitlement and encouraging feelings of equity.

2. Supply avenues so developers can express concerns and work style preference to the IDE creators. Paraphrase back concern and ask developer for input.
3. Conduct usability studies on features to ensure clear communication, reduce any unnecessary steps and test intrinsic motivation score as well as ensure structured work-flow, and optimal challenge.
4. Provide a means for developers to collaborate in solving problems and supply easy channels to find information about their problem.

While these four approaches cover all constructs of SDT, they focus heavily on feelings of relatedness. Since this was indicated in our survey as a large gap between intrinsic motivation and IDEs, we recommend primarily implementing features that enhance positive interactions amongst developers as well as trust and understanding of teammates code to elicit impactful change.

Finally, we would like to discuss the limitations of this study. Our conclusions could have been greatly strengthened by a larger sample set. While the group of developers surveyed was diverse, larger numbers of developers in multiple countries would have allowed us to make conclusions about the general development community with more certainty.

9. Future Work

Verifying that SDT provides an adequate lens to view developer motivation has opened the door to a number of new studies. It gives the computer science community a common foundation to discuss and study developer motivation. The survey tool can be used to test infinite features in any virtual tool. There seems to be a lot that could be learned by examining intrinsic motivation in OSS projects specific to the developer role and involvement. Based on our study and the importance of relatedness, testing community-based prototypes could aid in understanding the deficiencies in intrinsic motivation support in IDEs. Finally creating environments that enhance intrinsic motivation in end-user programmers could have a significantly positive impact on innovation in ubiquitous computing.

10. Conclusions

In order to fully support traits that promote exceptional software engineering we must examine the role of intrinsic motivation. SDT provides us with an appropriate framework to do so. The survey tool we generated offers a method to measure intrinsic motivation support in a person while using an IDE. Based on results from this tool, we determined that developers who contribute to OSS projects feel the highest level of intrinsic motivation. We also found that feelings of relatedness in IDEs are the area most lacking and with the highest impact potential for improving intrinsic motivation in end-user programmers and traditional developers.

11. REFERENCES

[1] ABET, “Criteria for Accrediting Engineering Programs 2012-2013.” http://www.abet.org/DisplayTemplates/DocsHandbook.aspx?id=3143

[2] Abramson, L.Y Seligman, M.E.P. & Teasdale, J.D. (1978). Learned helplessness in humans: Critique and reformulation. Ournal of Abnormal Psychology, 87, 49-74

[3] Amabile, T. M. (1996). Creativity and innovation in organizations. Harvard Business School.

[4] Bandura, A. (1986). Social foundations of thought and action: A social cognitive theory. Englewood Cliffs, NJ: Prentice-Hall.

[5] Baumeister, R., & Leary, M. R. (1995). The need to belong: Desire for inter personal attachments as a fundamental human motivation. Psychological Bulletin, 117, 497-529.

[6] Beckwith, L., Kissinger, C., Burnett, M., Wiedenbeck, S., Lawrance, J., Blackwell, A., & Cook, C. (2006, April). Tinkering and gender in end-user programmers' debugging. In Proceedings of the SIGCHI conference on Human Factors in computing systems (pp. 231-240). ACM.

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

[8] Bitzer, J., Schrettl, W., & Schröder, P. J. (2007). Intrinsic motivation in open source software development. Journal of Comparative Economics, 35(1), 160-169.

[9] Bowlby, J. (2012). The making and breaking of affectional bonds. Routledge.

[10] Blau, P. (1964). Exchange and power in social life. New York: Wiley.

[11] Chen, K. C., & Jang, S. J. (2010). Motivation in online learning: Testing a model of self-determination theory. Computers in Human Behavior, 26(4), 741-752.

[12] Chintakovid, T. (2009). Effects of gender, intrinsic motivation, and user perceptions in end-user applications at work (Doctoral dissertation, Drexel University).

[13] Crocker, J., & Canevello, A. (2008). Creating and undermining social support in communal relationships: The role of compassionate and self-image goals. Journal of Personality and Social Psychology, 95(3), 555-575.

[14] deCharms, R. (1968). Personal causation. New York: Academic Press

[15] Deci, E. L., Koestner, R., & Ryan, R. M. (1999). A meta-analytic review of experiments examining the effects of extrinsic rewards on intrinsic motivation. Psychological bulletin, 125(6), 627.

[16] Deci, E. L., & Ryan, R. M. (1991, August). A motivational approach to self: Integration in personality. In Nebraska symposium on motivation (Vol. 38, pp. 237-288).

[17] Deci, E. L. (1975). Intrinsic motivation. New York: Plenum.

[18] Deci, E. L. (1972). Intrinsic motivation, extrinsic reinforcement, and inequity. Journal of personality and social psychology, 22(1), 113-120.

[19] Deci, E. L., & Ryan, R. M. (1985). Self-Determination. John Wiley & Sons, Inc.

[20] Deci, E. L. (1980). The psychology of self-determination. Lexington, MA: Lexington Books

[21] Deci, E. L., & Ryan, R. M. (2000). The" what" and" why" of goal pursuits: Human needs and the self-determination of behavior. Psychological inquiry, 11(4), 227-268.

[22] Fisher, C. D. (1978). The effects of personal control, competence, and extrinsic reward systems on intrinsic motivation. Organizational Behavior and Human Performance, 21, 273-288.

[23] França, A. C. C., Gouveia, T. B., Santos, P. C., Santana, C. A., & da Silva, F. Q. (2011, April). Motivation in software engineering: A systematic review update. In Evaluation & Assessment in Software Engineering (EASE 2011), 15th Annual Conference on (pp. 154-163). IET.

[24] França, A. C. C., Carneiro, D. E., & Silva, F. Q. D. (2012, September). Towards an Explanatory Theory of Motivation in Software Engineering: A Qualitative Case Study of a Small Software Company. In Software Engineering (SBES), 2012 26th Brazilian Symposium on (pp. 61-70). IEEE.

[25] Garvin, J. (Feb 2013) "Demographics shed light on developer personalities" http://www.datanami.com/datanami/20130214/demographics_shed_light_on_the_programmer_personality.html

[26] Glucksberg, S. (1964). Problem solving: Response competition and the influence of drive. Psychological Reports, 15(3), 939-942

[27] Gouldner, A.W. (1960). The norm of reciprocity. American Sociological Review, 25, 161-178.

[28] Hars, A., & Ou, S. (2002). Working for free? Motivations for participating in open-source projects. International Journal of Electronic Commerce, 6, 25-40.

[29] Harter, S. (2009). Effectance motivation reconsidered. Toward a developmental model. Human development, 21(1), 34-64.

[30] Holmes, J.G. (1981). The exchange process in close relationships: Microbehavior and macro-motives. In M.J. Lerner & S.C. Lerner (Eds.) The justice motive in social behavior (pp.261-284). New York: Plenum.

[31] Kim, Y.., Butzel, J. S., & Ryan, R. M. (1998, June). Interdependence and well-being: A function of culture and relatedness needs. Paper presented at The International Society for the Study of Personal Relationships, Saratoga Spring, NY.

[32] Li, Y., Tan, C. H., Xu, H., & Teo, H. H. (2011). Open source software adoption: motivations of adopters and amotivations of non-adopters. ACM SIGMIS Database, 42(2), 76-94.

[33] Monthly Labor Review, "Occpuational employment projections to 2010 http://www.bls.gov/opub/mlr/2001/11/art4full.pdf

[34] Osterweil, L. (1981). Software environment research: Directions for the next five years. Computer, 14(4), 35-43.

[35] Osterweil, L. J. (1983). Toolpack—An experimental software development environment research project. Software Engineering, IEEE Transactions on, (6), 673-685.

[36] PR WEB, "Global Software Population to increase to 20 million by 2015,"http://www.prweb.com/releases/2011/9/prweb8834228.htm

[37] Ryan, R. M., & Frederick, C. M. (1997). On energy, personality, and health: Subjective vitality as a dynamic reflection of well-being. Journal of Personality, 65, 529-565.

[38] White, R. W. (1963). Ego and reality in psychoanalytic theory. New York: International Universities Press.

[39] Ryan, R. M. (1982). Control and information in the intrapersonal sphere: An extension of cognitive evaluation theory. Journal of Personality and Social Psychology, 43, 450-461.

[40] Ryan, R. M., & Connell, J. P. (1989). Perceived locus of causality and internalization: Examining reasons for acting in two domains. Journal of personality and social psychology, 57(5), 749-761.

[41] Ryan, R. M., Deci, E. L., & Grolnick, W. S. (1995). Autonomy, relatedness, and the self: Their relation to development and psychopathology.. Autonomy, relatedness, and the self: Their relation to development and psychopathology. In D. Cicchetti & D, J. Cohen (Eds.), Developmental psychopathology: Theory and methods (pp. 618-655). New York: Wiley

[42] Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American psychologist, 55(1), 68-78.

[43] Ryan, R. M., Kuhl, J., & Deci, E. L. (1997). Nature and autonomy: Organizational view of social and neurobiological aspects of self- regulation in behavior and development. Development and Psychopathology, 9, 701-728.

[44] Ryan, R. M., Rigby, C. S., & Przybylski, A. (2006). The motivational pull of video games: A self-determination theory approach. Motivation and Emotion, 30(4), 344-360.

[45] Reis, H. T. (1994). Domains of experience: Investigating relationship processes from three perspectives. In R. Erber & R. Gilmour (Eds.), Theoretical frameworks for personal relationships (pp. 87-110). Hills- dale, NJ: Erlbaum.

[46] Scaffidi, C., Shaw, M., & Myers, B. (2005, September). Estimating the numbers of end-users and end-user programmers. In Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on (pp. 207-214). IEEE.

[47] Seligman, M. E. P. (1975). Helplessness. San Francisco: Freeman

[48] Sharp, H., Baddoo, N., Beecham, S., Hall, T., & Robinson, H. (2009). Models of motivation in software engineering. Information and Software Technology, 51(1), 219-233.

[49] Sheldon, K. M., Ryan, R. M., Rawsthorne, L. J., & Ilardi, B. (1997). Trait self and true self: Cross-role variation in the Big-Five personality traits and its relations with psychological authenticity and subjective well-being. Journal of Personality and Social Psychology, 73, 1380-1393.

[50] Shneiderman, B. (2007). Creativity support tools: accelerating discovery and innovation. Communications of the ACM, 50(12), 20-32.

[51] Skinner, B. F. (1953). Science and human behavior. New York: Macmillan.

[52] Steinberg, L., & Silverberg, S. (1986). The vicissitudes of autonomy in adolescence. Child Development, 57, 841-851.

[53] Turley, R. T., & Bieman, J. M. (1995). Competencies of exceptional and nonexceptional software engineers. Journal of Systems and Software, 28(1), 19-38.

[54] Vallerand, R. J. (1997). Toward a hierarchical model of intrinsic and extrinsic motivation. Advances in experimental social psychology, 29, 271-360.

[55] Waterman, A. S. (1993). Two conceptions of happiness: Contrasts of personal expressiveness and hedonic enjoyment. Journal of Personality and Social Psychology, 64, 678- 691.

[56] Weisberg, R. W. (2006). Creativity: Understanding innovation in problem solving, science, invention, and the arts. Wiley.

[57] Wehmeyer, M. L. (1992). Self-determination and the education of students with mental retardation. Education and Training in Mental Retardation, 27, 302-314.

[58] Wehmeyer, M. L., & Berkobien, R. (1991). Selfdetermination and self-advocacy: A case of mistaken identity. The Association for Persons with Severe Handicaps Newsletter, 7, 4.

[59] Deci, E. L., Ryan, R. M., Gagné, M., Leone, D. R., Usunov, J., & Kornazheva, B. P. (2001). Need satisfaction, motivation, and well-being in the work organizations of a former Eastern Bloc country. Personality and Social Psychology Bulletin, in press.

[60] La Guardia, J. G., Ryan, R. M., Couchman, C. E., & Deci, E. L. (2000). Within-person variation in security of attachment: A self-determination theory perspective on attachment, need fulfillment, and well-being. Journal of Personality and Social Psychology, 79, 367-384.

[61] Ilardi, B. C., Leone, D., Kasser, R., & Ryan, R. M. (1993). Employee and supervisor ratings of motivation: Main effects and discrepancies associated with job satisfaction and adjustment in a factory setting. Journal of Applied Social Psychology, 23, 1789-1805.

[62] Kasser, T., Davey, J., & Ryan, R. M. (1992). Motivation, dependability, and employee-supervisor discrepancies in psychiatric vocational rehabilitation settings. Rehabilitation Psychology, 37, 175-187.

An Evaluation of Game Controllers and Tablets as Controllers for Interactive TV Applications

Dale Cox, Justin Wolford, Dedrie Beardsley and Carlos Jensen

11.1 ABSTRACT

There is a growing interest in bringing online and streaming content to the television. Gaming platforms such as the PS3, Xbox 360 and Wii are at the center of this digital convergence; platforms for accessing new media services. This presents a number of interface challenges, as controllers designed for gaming have to be adapted to accessing online content. This paper presents a user study examining the limitations and affordances of novel game controllers in an interactive TV (iTV) context and compares them to "second display" approaches using tablets. We look at task completion times, accuracy and user satisfaction across a number of tasks and find that the Wiimote is most liked and performed best in almost all tasks. Participants found the Kinect difficult to use, which led to slow performance and high error rates. We discuss challenges and opportunities for the future convergence of game consoles and iTV.

12. INTRODUCTION

Interactive television promises to give viewers more flexibility and control over their viewing experience, while enriching it with a wealth of Internet accessible content and information. By giving the viewer a communication channel back to service providers, viewers can not just access new services, but also shape and control their viewing experience in ways that were not possible before. This has so far led to the emergence of services such as video on demand (Netflix, Hulu, etc.),the presentation of Internet content on TV screens (YouTube & Flickr channels on Apple TV, etc.), but could also allow for content ratings, interactive or contextual searching or social networking (see Boxee.tv).

For trivial tasks like navigating a simple movie rental UI, or controlling streaming video content, a traditional remote control is often sufficient. For instance, the Apple TV remote control is among the simplest available, with 7 buttons. However, as services become more complex and rich, requiring more or finer levels of control or interaction – such as navigating a non-trivial web page or GUI, carrying out drag and drop tasks, or more extensive text entry for search or socialization – more sophisticated input devices may be required. Some of these tasks overlap with those integral to the modern gaming experience. Video game consoles, with dedicated game controllers, due to their pervasiveness, connectivity and processing power are often at the center of this digital convergence of TV and Internet content. It is therefore important to examine how suitable current systems are for bridging this gap.

The last generation of game consoles have each introduced some device capable of spatial gestures allowing the possibility for a natural user interface (NUI), and which could be especially helpful for navigating complex UIs. Microsoft released the Kinect for the Xbox 360 in 2010, a camera based system that tracks players' movement to allow for complex and natural interactions

without holding any kind of controller. Nintendo released the Wii in 2006, which introduced the Wii Remote (nicknamed Wiimote), a wireless controller that tracks spatial movement through accelerometers and infrared sensors. Sony has a similar system for the Playstation 3.

These game consoles and their controllers have sparked the development of tools and solutions beyond those in traditional gaming. The motion-driven Nintendo Wiimote was the first to attract the attention of the hacker community outside the console market. The Bluetooth interface of the Nintendo Wiimote made it a simple and accessible device to "hack" and adopt for various uses. Soon after the Kinect was released, the open source community reverse engineered the device and released a driver package allowing others to develop systems that took advantage of its capabilities. Its USB interface makes it ideal for use with a PC. Today, several different open source SDKs exist, as well as an official Microsoft Kinect SDK for Windows.

In addition, tablets continue to evolve and increase in popularity. Some proposed game controllers are exploring the use of such touch interfaces (most notably the upcoming Wii U). Tablets are also being considered as companions to both gaming devices and iTV services (often referred to as second screen navigation). According to a recent Pew study [7], tablet ownership nearly doubled over just the 2011 holiday season. There are a number of applications currently available that allow a tablet to serve as an input device for another computer such as IntoNow (http://www.intonow.com) that detects which show or movie you're watching and provides additional media content and social networking capabilities. These applications allow the tablet to function like a touch screen display or to track pad found on most laptops and would serve as a suitable baseline.

This paper explores the challenges and opportunities of using game-related control technologies to control interactive television applications through

the study of a hypothetical, but representative set of navigation, selection, and control tasks for a iTV application. We examine learnability and ease of use, as well as accuracy and error rates.

The rest of this paper is laid out as follows. First we discuss related work looking at the control of interactive television applications as well as gaming systems. We then discuss our user experiment and the design considerations we took into account. Finally, we present our findings, and a discussion of future work.

13. RELATED WORK

Much innovation has taken place in the design of new interaction techniques and devices for gaming devices. The research community is still catching up with the necessary evaluation of the potential, effectiveness and usability of these novel game input devices, both within their targeted use domain as well as in other environments like PC or interactive television.

Looking to the modern interactive TV interface, we see that it combines many types of user interaction. The areas we chose to focus on cover the core functionality of pointing, navigation, and text entry. Pointing is perhaps the more novel and difficult task with todays' hardware, but is a prerequisite for many of the more sophisticated types of applications and use cases. Individually, each of these topics has an established body of research, but in the context of a media center or iTV there is very little. We also present the recent adoption rate history of streaming services, which are at the center of modern interactive television systems.

13.1 Growing Popularity of Streaming Services

With the spread of broadband internet access throughout North America, high bandwidth services like high definition on-demand streaming video, previously limited in either quality or duration, have become commonplace. Between 2001 and 2009, broadband Internet use increased seven-fold, covering from 9% to 64% of American households [21]. A 2011 Nielsen study found that from 2008 to 2011 there was a 22% increase in the number of users watching video on the Internet, and an 80% increase in the average viewing time [19].

One of the key players in the Internet-based video-on-demand area has been Netflix. Netflix debuted as a DVD-by-mail service in 1997, and has since introduced and popularized a broadly available Internet streaming service. By the

end of 2011, Netflix had over 21 million paying streaming subscribers [13]. In a Fall 2010 report by Sandvine, Netflix was shown to account for 20.6% of all downstream prime-time Internet traffic in North America [17]. Just 7 months later, Netflix users were consuming 29.7% of all downstream prime-time Internet traffic in North America [17].

Other online video services such as Hulu, Amazon Instant Video and YouTube (though the latter still mostly offers shorter clips, it has branched into feature content delivery as well) have also grown in popularity. Internationally, over 4 billion videos are viewed on YouTube each day [12]. Hulu just passed 1.5 million paying subscribers of its paid Plus service [18].

In part this success is driven by the growth of systems that help these users bridge the gap between the computer and the TV experience. This includes a plethora of streaming devices like the Roku and Apple TV, a new generation of connected TV's and DVD/Blue-ray players, and last but not least game consoles. Each of the three leading game consoles have added mechanisms for viewing streaming Netflix content on their devices. Services like Netflix and Hulu that began as a PC experiences, can now be accessed from a number of different devices and platforms. This has made enabled these services to go from a niche technophile market to appealing to the average consumer. In a 2011 Nielsen study, 50% of all Netflix users were found to watch Netflix content through a gaming console [16]. In the same study, Nielsen found 162 million Americans own a game console. This means that these platforms are natural ways to deliver these experiences. The need to manage users' media viewing experience has led to the development of media center applications like the Xbox Media Center (XBMC) and Windows Media Center. Internet-enabled set-top devices like Boxee and AppleTV have also appeared allowing easy streaming video viewing from a normal TV. These allow users to consolidate their media consumption, as well as manage their local library. Due to the interactive and highly customizable

experience allowed by these services, the need for robust input methods will continue to gain importance.

13.2 Pointing and Navigation Using Novel UI Devices

Over the last few years there has been a growing trend to develop and evaluate what are being referred to as Natural User Interfaces (NUI's). These interfaces extend the basic direct manipulation paradigm by allowing users to interact with the computer with motions more closely resembling those we'd use in real life. Among the leading platforms for such interfaces we find game consoles. These techniques could help bridge the complexity gap between the new interactive TV applications and the interactions afforded by conventional remote controls. Because of space limitations we will only review some of the most directly applicable research to our study.

Starting with camera and motion based techniques, Cheng and Takatsuka [2] introduced dTouch, a finger pointing technique for large displays that uses an off-the-shelf webcam. Using the concept of a "virtual touchscreen", dTouch enables users to manipulate onscreen objects in an absolute coordinate system. They performed a user study comparing dTouch to a method using the EyeToy camera, used on the PlayStation console. Results indicated the two methods were comparable with users preferring dTouch.

Lee [3] described a cursor technique using the Wiimote that enabled finger-tracking through the use of reflective tags taped to the fingers of users. Rather than holding the Wiimote in the hand, they used the IR camera built into the Wiimote with an IR LED array to allow almost bare-hand operation. Lin et al. [4] demonstrated a technique similar to Lee's, but using a second Wiimote for additional functionality.

Using more traditional controllers, Natapov et al. [5] performed a

comparative study evaluating the Wiimote and traditional gamepad for pointing and selecting tasks. Although the error rate was higher, 14 out of 15 participants said they preferred the Wiimote in a home entertainment environment. They found that the Wiimote had a 75% performance increase over the traditional gamepad when comparing speed and accuracy.

Finally, turning to smart phones and tablets, McCallum et al. [10] developed a hybrid system called ARC-Pad, which combined absolute and relative positioning techniques for use with large displays. A smart phone screen was used like a touchpad. ARC-Pad was compared against a traditional touchpad style interface, which employed cursor acceleration. ARC-Pad performed slightly better (166ms faster) than the relative in completion time. The results suggested as pixel distance increased beyond what was studied, ARC-Pad performance would change minimally while the relative touchpad would continue to worsen.

13.3 Text Entry With Keyboard Alternatives

Over the last two decades, the need for text entry without a traditional mechanical keyboard has increased. With the introduction of PDAs and smart phones, text entry presents a challenge due to a limited input area. Most interactive TV systems attempt to minimize the necessity of text entry through the use of various widgets and interface choices. Though the need may be reduced, it is difficult to completely do away with text entry for applications such as search or social media.

This has led TV manufacturers like Samsung to market 2-sided remote controls; one side having normal remote control functions and the reverse a full keyboard, or Sony to merge a PlayStation controller and a full keyboard in their Google TV products. While such solutions may provide speed advantages, they lead to cumbersome and intimidating user experiences. We examined alternatives

to keyboard text entry, focusing on touchscreens, game controllers and freehand gesture techniques.

The Graffiti pen-based gesture alphabet was made popular by Palm in the late 90s. It allowed users to quickly input text using a proprietary alphabet. MacKenzie and Zhang [9] conducted a study analyzing the learnability and accuracy of Graffiti. Participants were given practice time using a reference chart showing the gesture alphabet. After practice they repeated the entire alphabet 5 times without having a reference available and again 1 week later. The results showed a nearly 97% character accuracy rate after 5 minutes of practice.

Tao, et al. [11] adapted the Graffiti alphabet to a freehand gesture-based text entry system called AirStroke. A user study was performed comparing two AirStroke implementations, one with word completion and one without. Participants completed 20 sessions each, over a period of two weeks in which error-rates and speed were recorded. Airstroke with word completion averaged 11 wpm while no word completion was at 6.5 wpm. The error rate with word completion averaged 6.6% compared to 11.8% without. Some participants initially reported arm fatigue, which lessened as their proficiency increased.

Several techniques have been developed enabling text input using a traditional gamepad. Költringer et al. [15] designed and evaluated TwoStick, a novel text entry system using both analog joysticks on an Xbox 360 controller. TwoStick was compared to a traditional selection keyboard. Initially, users typed slower and had a higher error rate using TwoStick, but after 15 sessions TwoStick averaged 14.87 wpm while the selection keyboard had a mean of 12.9 wpm. Wilson and Agrawala [14] also created a dual joystick QWERTY method, which showed modest improvement upon the traditional single stick selection keyboard.

Shoemaker et al. [8] compared 3 techniques for mid-air text input. A circle keyboard, QWERTY keyboard and cube keyboard all used a Wiimote as an input method. The QWERTY method performed best in accuracy and performance; this

method is similar to our Wiimote text entry task. A questionnaire taken after the study revealed users preferred the QWERTY method overall.

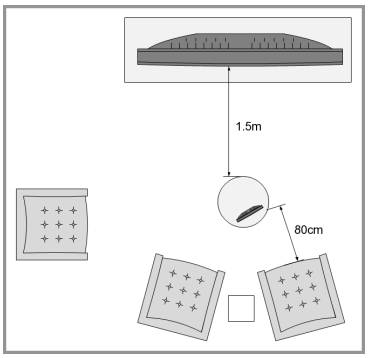
Castellucci and MacKenzie [1] presented an alternative to an on-screen keyboard using the Wiimote called UniGest. UniGest is a technique that takes advantage of the motion-sensing capabilities of the controller to capture movement and rotation. A gesture alphabet is proposed which maps the gestures to character input. Their results predict an upper-bound of 27.9 wpm using the UniGest technique.

14. Methodology

This section describes a user study designed to measure the effectiveness of video game and tablet input methods in a iTV context. We used 4 input methods: the Microsoft Kinect, Nintendo Wiimote and 2 methods using an Android tablet; a condition where subjects had to scroll (relative coordinate condition), and one using an absolute coordinate space (mirror condition). The idea was that in the relative condition subjects would have to scroll around like when using a mouse pad, and in the absolute coordinate condition, the whole TV image would be represented on tablet at once. Participants completed pre and post-experiment questionnaires and also a post-experiment interview. All sessions were recorded using a video camera and screen capturing software.

All participants were recruited in pairs from a college campus and surrounding community. There were a total of 62 participants, 33 male and 29 female. All but 4 were right-handed. Their ages ranged from 18 to 57 years old with a mean of 24.5. Prior to the experiment, participants completed a questionnaire gathering demographic data and media viewing frequency. Each pair of participants was assigned 2 devices to use, and all device pair permutations were assigned randomly.

The system ran on a PC hooked up to a 55" HDTV, set up in an environment designed to look and feel like a living room (see room layout in Figure 1). The subjects sat in the two center seats, while the experimenter sat off to the side with a good view of the subjects. The table in the middle was positioned far enough away that subjects could not use it to hold items while performing their task. There was a small table (15x15cm surface area) between the two chairs, large enough to hold a drink or a plate, but not both at the same time.



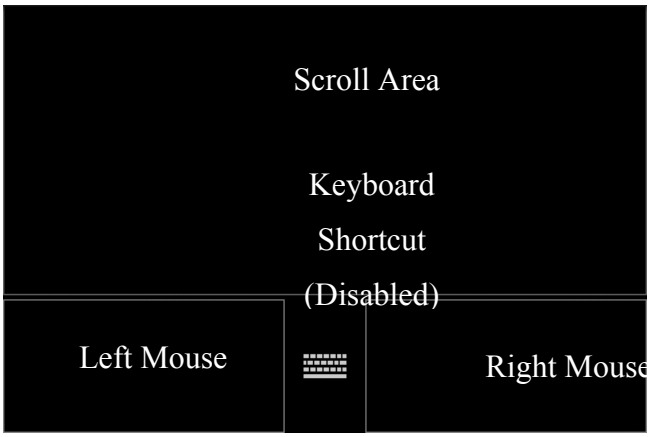
- Figure 3: Room configuration used during the experiment.

A Nintendo wireless sensor bar was used in combination with a standard Nintendo Wii Remote for relevant conditions. A Microsoft Kinect sensor was used for the Kinect tasks, and a 10.1" tablet running Android 2.3 was used for the tablet tasks. A windows application called GlovePIE was used to control the cursor using the Wiimote. A GlovePIE script enabled the IR camera in the Wiimote to control the mouse cursor and the 'A' button to control the left mouse click.

A custom application was created to allow the Kinect to control the mouse cursor and left button. The application was written in C# using the OpenNI framework. To move the cursor, participants moved their right hand, which positioned the cursor similar to a traditional mouse. To initiate a drag, participants moved their left hand forward to cross the threshold of a virtual plane 30-40cm in front of them. This action is equivalent to a left mouse down event. To initiate a drop, participants would simply pull their arm back and break the plane in the opposite direction. This action is equivalent to a left mouse up event. To initiate a left click, participants move their left hand quickly through the plane and back out in one fluid motion.

The software used in the relative tablet condition was an open source Android application called RemoteDroid. This application turns the entire tablet

into one large touchpad similar to what is found on most laptop computers (see Figure 4). This is application was paired with an application that runs on the host computer.



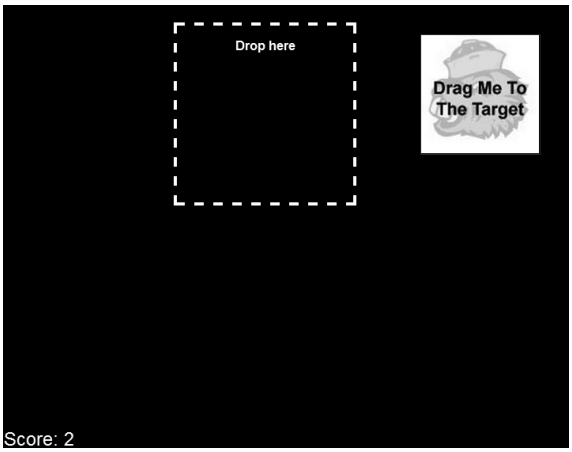
- Figure 4: The configuration of the relative tablet app.

The mirrored condition used a modified version of the RemoteDroid application. It continually updates the tablet display with a screenshot of the TV. Instead of using a relative coordinate system where the cursor movement corresponds to relative changes in cursor position, an absolute coordinate system is used. By using an absolute coordinate system, a user can click on any location of the mirrored display and have it mapped to the equivalent location on the PC display.

Subjects were trained on the devices they were going to use and given time to practice on a screen that allowed dragging and dropping an object and clicking a button. When they felt comfortable with the device they began the drag-and-drop task.

The only actions allowed in the drag and drop task were dragging and dropping a widget into a target box (see Figure 3). If the widget was dropped fully within the boundary of a target presented at a random point on the screen, then a hit was recorded, and the subject would be presented with a new, slightly smaller

target. A miss was recorded if the user missed the widget when attempting to select it, or if they released the widget outside of the target box. They were able to keep trying until they ran out of time for the trial. If the user was unable to place the widget in the target within 16 seconds, the box and widget were moved to random locations on the screen and the target box got bigger. If the user hit the target, then both the widget and target were randomly moved and the target shrank, with the minimum size for the target being 3 pixels wider and taller than the widget.



- Figure 5: Sample screen for drag and drop task.

After completing the drag-and-drop accuracy task, we asked subjects to complete a number of navigation tasks on an iTV environment, simulated using the popular XBMC media center.

The navigation tasks included 3 different activities in XBMC. The first required the subject to navigate from the main menu to the weather settings screen and change the city name. This may have been the most challenging task due to the text entry requirement. Next, subjects would go to the weather screen and change the city currently displayed. This was difficult at times because it required

clicking on a very small button. After changing the city, the subject would navigate to the movie selection screen and select a movie using a scrollbar. Finally, after the movie started, a slider was used to adjust movie volume. In all navigation tasks, an error was recorded if a subject clicked on a non-interactive item or if they clicked on the wrong UI widget.

We measured users' time to complete tasks and their error rates. Additionally we collected qualitative data in a post experiment interview and survey. Finally we used screen capture software and a video camera to record subjects. Subjects performed the experiment in pairs, taking turns with each device (subject A would try device 1, then subject B would use the same device. Next Subject A would try device 2, and then Subject B would do the same). Pairs were randomly assigned two input methods.

Because of previous research showing the importance of studying the effectiveness of UI techniques under similar manual loads [20], and the oft-informal nature of TV viewing, we decided to give each subject a slice of pizza and a drink to hold and consume during the course of the experimental tasks. Subjects were not allowed to place the food items on the floor or on the larger central table, but had to balance them on their seat or lap.

After both subjects completed all tasks using the first device assigned to them, they were introduced to the second device, and the process started anew, from the training period onward. After both subjects completed both conditions, they were asked to complete a short survey asking them about learnability, ease of use and practicality of the devices they had been assigned. All questions were on a 5-point Likert scale, 1 meaning strongly agree and 5 meaning strongly disagree.

Finally, they were interviewed to get a deeper understanding of their experience. We were interested in their satisfaction with the various input devices. This included the ease with which the subjects could use the device along with

their enjoyment of using the device. Additionally we asked about their comfort level using the device in a social context where others were observing them.

15. RESULTS

15.1 Drag and Drop Task

The main task we used to measure the efficiency of a UI technique for manipulation was the timed drag and drop task, as it combined selection, movement, as well as accuracy. The more drops a subject managed within the time allotted, the more accurate their manipulation of the widgets on the screen. The highest mean number of targets hit was with the mirror tablet, where subjects hit an average of 14.09 targets (see Figure 4). Subjects using the Wiimote and relative tablet scored 12.97 and 11.90 hits respectively. Those using the Kinect averaged a score of 7.37 hits. The Kinect did significantly worse than all other devices (One-way ANOVA $F(3,19)=54.5, P<0.001$ with Tukey's HSD for Post Hoc analysis). The relative tablet also did significantly worse than the Wiimote and mirror tablet ($P<0.05$). There was no significant difference between the Wiimote and mirror devices.

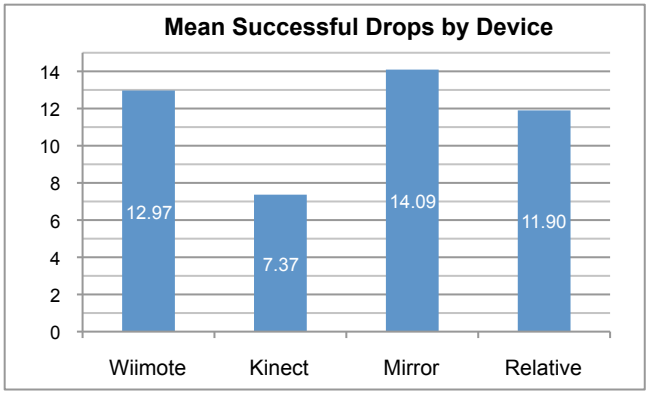


Figure 6: More drops show that the user was quicker and more accurate than users with fewer successful drops.

There is of course a direct relation between accuracy and speed in this task. The quicker the manipulation, the more likely you are to be able to complete the task, and even try multiple times in case of failure. Therefore an inaccurate but very quick technique could lead to misleading results. To investigate this we decided to look at the average target size for the last 5 targets subjects successfully hit. This allowed us to give subjects some additional practice time, and allowed subjects' performance to plateau. The results are shown in figure 5.

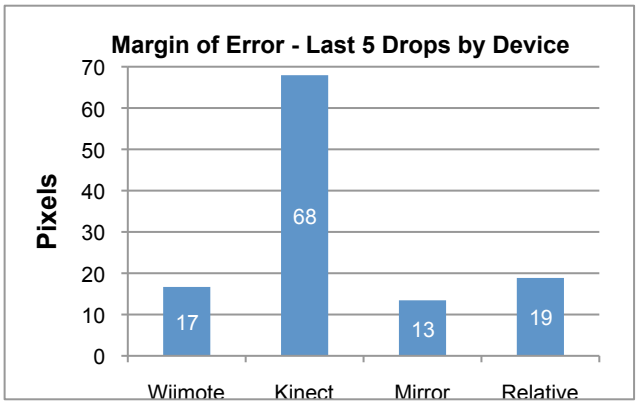


Figure 7: Average final margin of error in pixels by condition

Users were much less accurate with the Kinect than with any other device. Users of the mirror, relative and Wiimote conditions averaged a 13 to 19-pixel difference between the widget dimensions and the target dimensions. With the widget being 152px square, this meant a margin of error of less than ±9-12% of the widget size, or ±1-2% of the total screen real estate (1920x1080). For the Kinect the margin of error was ±45% of the widget, and ±6% of the total screen real estate. There were significant differences (One-way ANOVA $F(3,120)=13.77,\ P<0.001$ with Tukey's HSD for Post Hoc analysis), the Kinect

was significantly different from the other devices ($P<0.001$). Other differences were not significant.

During the experiment and in the post experiment interview, several subjects mentioned that the sensitivity for the relative tablet was low and that they would have to slide their finger across the device more than once to get the cursor to traverse from one side of the screen to the other. Users needed to swipe 3 times to go from one side of the screen to the other. No enhancements such as cursor acceleration were implemented; this could potentially improve performance of the relative condition. This may in part explain why the relative tablet scored worse than the mirror and Wiimote. However, because speed and accuracy are often traded off against each other, it is not a given that acceleration would lead to better results. This is something that should be investigated more in-depth. Users were not able to move the cursor rapidly enough to hit the same number of targets.

The issues with using the Kinect were more pronounced and deep-rooted. Subjects were observed having a difficult time both beginning a drag (selecting and dragging the target) and dropping the widget into the target (widget would often be dropped prematurely and unintentionally). A less common but also real problem was that in order to establish a difference between a click event and a drag event users had to press forward and hold for 0.5 seconds before beginning the drag. It was common to see users attempt to drag before the drag event had been registered. They were told about this in the training but as the user began the trial and were trying to rush through the task, they would often not pause long enough. Some visual indicator to let them know that the event had been registered could have made a difference.

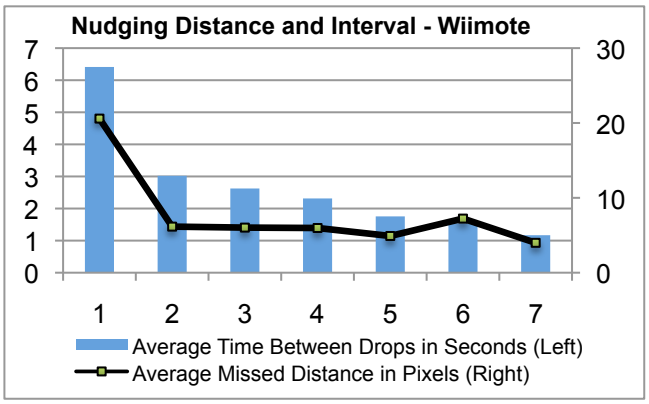
The more fundamental problem with the Kinect condition was the 2-handed operations. Subjects usually had little trouble placing a cursor over a target using one hand, though fatigue was mentioned as a concern in some trials.

However, the action of bringing or removing the second hand from the camera plane often caused subjects to inadvertently rotate their bodies to retain balance, even while seated. This of course would make their targeting hand move, resulting in a missed target. This same phenomenon was observed time and again across tasks and subjects. The only effective remedy we saw was for subjects to plant their elbows in the seat, and use this to counter the natural body rotation action. Though effective, this led to a very restrictive seating position.

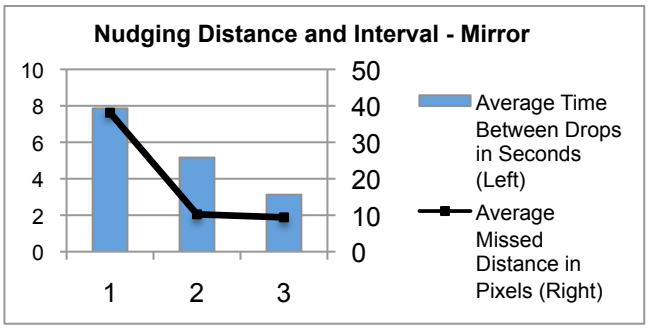
15.2 Nudging

There is a tradeoff between speed and accuracy, and with a sufficiently fast UI, users can home in on the target effectively. We referred to this behavior as “nudging”. In our experiment, this turned out to be a relatively common strategy; if a subject failed to hit the target on the first try, they would rethink their strategy (a longer pause) and then pick up the widget and home in through a series of rapid follow-up moves. This was especially common with the smaller targets, where the margin for error was low. The majority of times subject were able to hit the target in one or two attempts. However, some times it took longer.

16. Figure 6 shows how subjects using the Wiimote employed this strategy. Wiimote users were among the most successful and accurate, and the technique allowed for quick and easy nudging, or homing in on the target. As we can see, after a longer rethink following an initial miss, subjects engaged in a lot of rapid moves aimed at trying to hit the target. Subjects in this condition were still among the most accurate and successful. We see a very similar behavior among subjects using the mirror tablet application, though there is less of a long-tail (see Figure 7).



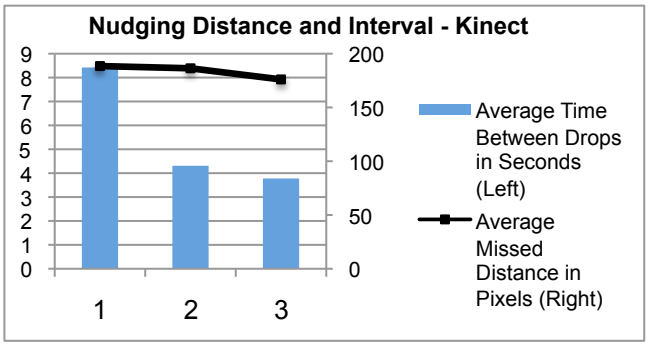
- Figure 8: Nudging interval (left axis, blue columns, in seconds) and distance (black line, right axis, in pixels) over number of tries to hit one target – Wiimote.



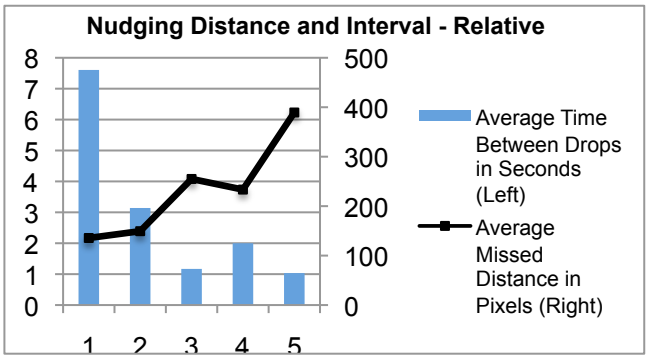
- Figure 9: Nudging interval (left axis, blue columns, in seconds) and distance (black line, right axis, in pixels) over number of tries to hit one target – Mirror Tablet.

This strategy however seemed to be less successful, even counterproductive in the other two conditions (see Figure 8 and Figure 9). In the case of the Kinect condition, accuracy was an enormous issue, and though subjects were more successful with repeated tries, they did not home in on the target, but rather hit random new points. In the case of the relative tablet application, the nudging strategy appears to be counterproductive. Subjects would after the second try engage in very rapid moves that rather than take them closer

to the target would distance them more. To us this is an important distinction between these two groups of techniques. Our subjects naturally gravitated to this strategy, and therefore it should be supported.



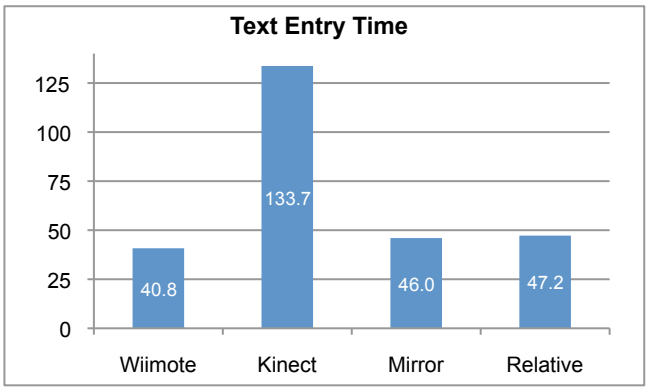
- Figure 10: Nudging interval (left axis, blue columns, in seconds) and distance (black line, right axis, in pixels) over number of tries to hit one target – Kinect.



- Figure 11: Nudging interval (left axis, blue columns, in seconds) and distance (black line, right axis, in pixels) over number of tries to hit one target – Relative Tablet.

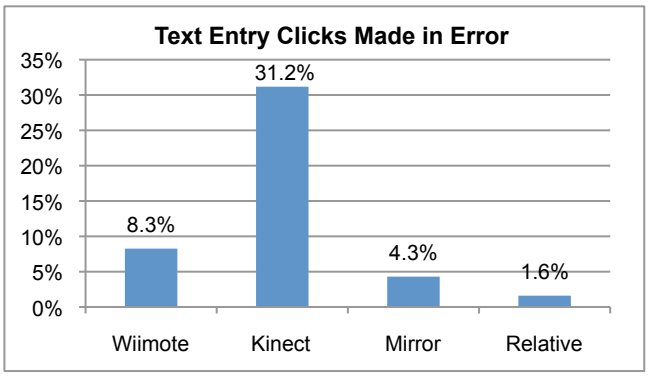
16.1 Text Entry

An important task for iTV applications is text entry, as it allows more rapid customization, search, etc. We chose to examine two factors, the number of clicks that landed off of the intended target (key) and the time it took users to input the text string (a 9 character string).



- Figure 12: Average text entry time in seconds

As we see in Figure 12, text entry was significantly slower with the Kinect when compared to the other devices (One-way ANOVA $F(3,115)=19.53$ $P<0.001$ with Tukey's HSD for Post Hoc analysis). The Kinect was significantly slower than all other devices ($P<0.001$). There were no other significant differences.



- Figure 13: Text entry mistakes using virtual keyboard

Again, because a lack of accuracy can lead to slower task completion, we chose to look at how many mistakes subject made. A mistake in this case could be a subject hitting the wrong letter, or trying to click on something other than a letter on the on-screen virtual keyboard. As in just about every task in our experiment, the Kinect fared most poorly with 31.2% of clicks missing their target. This is significantly worse than the other three devices (One-way ANOVA $F(3,116)=19.14$ $P<0.01$ with Tukey's HSD for Post Hoc analysis).

Of the remaining devices the Wiimote fared the worst with an error rate of 8.3%, though this did not affect completion time. The mirror app came next, with an error rate of 4.3%, likely caused by the small size of the keys when shown on the tablet. The error rate for the relative app was a surprisingly low 1.6%. The difference between these devices was not significant.

16.2 Effects of Prior Experience

Prior experience can have a significant impact on performance, especially when dealing with the novel. Subjects were asked to rate their prior experience with devices similar to those used in our study. A linear regression was used to compare the number of successful drops in the drag and drop task and the speed of text entry versus their prior experience (see Table 1).

- Table 16.1 Slope of linear models. Steeper slopes indicate stronger experience effect.

	Hits vs. Experience	Text Entry vs. Experience
Wiimote	0.61	0.33
Kinect	0.51	-5.42
Relative	0.39	-0.79
Mirror	0.25	-7.85

We see that the prior experience played the largest role in the Wiimote case. Despite being seen as universally easy to use, subjects were able to effectively leverage prior experience to improve further. The same was the case in the Kinect condition, though here, novices really suffered, and even experts performed marginally. There may have been a floor effect here as fewer subjects had experience with Kinect compared to other devices, and those that did have experience ended to have less experience than with the Kinect than with other devices.

More surprising, there was only a relatively mild learning effect for the two tablet solutions. While experience did help, it seems that these two techniques were so universally well-known and intuitive that all subject were able to use them effectively regardless of experience level.

17. Discussion

In our post experiment interviews we focused on understanding the limitations and advantages of the different approaches. One thing we stumbled on were issues related to the sensitivity, or lack thereof for some of our conditions. 76% of users or the relative tablet mentioned that sensitivity was an issue (it took too long to scroll from one side of the display to the other). 43% of Wiimote subjects complained about the device being too sensitive, reacting to slight hand tremors.

Despite the negative results, 13% of Kinect subjects commented positively about its usefulness. This was obviously surprising, but shows that people like the concept of this technique, if not the implementation. We also found that 40% of those who used the mirror tablet and 47% of those who used the Wiimote commented positively about these. Surprisingly only 13% of the relative tablet users commented positively about it despite the high performance.

When asked if the input method could be learned quickly, the Wiimote won out, and it was also rated as the least awkward to use. Subjects liked the simplicity of the Wiimote, both in interface navigation and physically. One participant said, "It was simple. Just point and click. You just aim at it and it's right there." One of the most common answers about what people liked about the Wiimote input method was that it had only 1 button. People also liked the familiarity with holding the Wiimote, that it felt like a remote control and had a physical button.By far, the least liked device was the Kinect. The most frequent negative comments had to do with physical fatigue and issues with sensor range and sensitivity. Having to hold their right arm up to position the cursor and left arm for click control resulted in almost all Kinect users complaining of arm fatigue. Finding a one-handed method for controlling the system could result in a significant improvement, as indicated by 60% of Kinect users. 60% also

complained about the sensor range or sensitivity. To limit interference, the sensor was placed 80cm away from the subject. As a consequence, subjects felt they were unable to move their hands as far to the left and right as they would like. Although most comments focused on why the Kinect was not effective, several participants liked how it did not require them to hold a physical device. One participant talked about how nice it would be to have no remotes and control everything with gestures. Most subjects however indicated that they would be embarrassed to use this technique in front of friends and family.

The two tablet techniques achieved roughly the same ratings, which were generally good. 30% of mirror application users commented positively on being able to directly manipulate the interface. One person said "I really liked being able to click on exactly what I can see on the screen." Others disagreed, saying how they prefer to only have one screen to interact with. One participant commented "Occasionally I found myself not knowing which screen to look at."

A side-effect of our implementation of the mirror app was noticeable "lag" between the TV and the tablet images of between 0.25 and 0.5 seconds. This delay was often mentioned as an annoyance. Likewise, nearly everyone who used the relative tablet app disliked its low sensitivity and the lack of acceleration techniques. With appropriate tweaking, both of these techniques would have likely scored higher on both likability and effectiveness.

18. Future Work

This was meant as an exploratory study examining the relative merits of a number of gaming-related UI methods, and their usefulness in an iTV setting. Looking forward, there are several improvements worth exploring based both on user feedback and our findings. As mentioned in the results and discussion sections, implementing motion smoothing for the Wiimote, acceleration for the relative tablet app, and reducing the lag for the mirror app are natural next steps. We believe all of these could drastically improve the user experience.

In implementing smoothing for the Wiimote, a slight delay will be introduced. Pavlovych and Stuerzlinger [22] studied the relationship between jitter and latency and their results could help inform an appropriate balance.

A more tricky problem was the noisiness and occasional false positive for hands for the Kinect. We were unable to use the Kinect API's native skeleton tracking because only the upper half of the users' body was visible to the sensor. Instead, we used a hand tracking method and filtered based on depth field data. Even with these precautions, a knee or other object could register as a hand. This led to a very frustrating user experience. In future revisions, we would look for a more robust tracking solution. We did not use the official Kinect SDK as it was not available in time.

When asked what they would change about the Kinect method, several people said they would prefer a one-handed solution. We think this would substantially decrease the physical fatigue and provide a more intuitive experience. Due to the Kinect's 640x480 resolution depth camera, robust finger tracking was difficult. Perhaps with a different library or algorithm, a more feasible approach could be found. One option might be the work of Oikonomidis et al. [6], who have demonstrated complex finger articulation, though not in real-time.

19. We chose not to investigate the use of voice commands in our experiment, in part because it would be difficult to filter noise and could be socially awkward. As the introduction of the Siri system on the iPhone, and the flurry of interest this has caused, these assumptions and prejudices may need to be revisited in future work.

20. Conclusions

With the growing availability of broadband Internet access, highly extensible game consoles, and the increasing popularity of social and streaming online entertainment services, their convergence is presenting a number of new challenges for HCI researchers. To the best of the authors' knowledge, there has been very little research on the adoption and use of novel game controller technology in a media center or iTV context. We hope our research will serve as a base for future work in this area.

Looking at the results we see that devices designed for gaming have the potential to be effective input devices for a typical iTV interface. We also see that some devices are better suited to this task than others. The Wiimote was effective, well liked, and very easy to learn. At the same time, it offered ample room for improvement as users gain experience. On the other hand, it is potentially limiting UI-wise, as all information has to be displayed on the primary display. The tablet systems were both well liked and effective as well, though they potentially offer more flexibility and exploration, albeit at a much higher hardware cost.

The Kinect, though appealing to many subjects due to its novelty and the promise of device-free interaction, proved to be too unreliable and cumbersome to use for any extended period of time. While it may be refined with better hardware and algorithms, its suitability and desirability for a social lean-back viewing experience may be limited. Fear of ridicule as much as physical fatigue and the problem of interference from others' movement are serious problems that need to be overcome.

21. REFERENCES

[1] S.J. Castellucci and I.S. MacKenzie. “Unigest: text entry using three degrees of motion” In extended abstracts of the 2008 Conf on Human factors in computing systems; CHI '08. ACM, New York, NY, USA, 3549-3554.

[2] K. Cheng and M. Takatsuka. “Initial evaluation of a bare-hand interaction technique for large displays using a webcam” In Proc. of the 1st ACM SIGCHI symposium on Engineering interactive computing systems (EICS '09). ACM, New York, NY, USA, 291-296.

[3] J. C. Lee. Hacking the Nintendo Wii Remote. IEEE Pervasive Computing, 7(3):39–45, July 2008.

[4] J. Lin, H. Nishino, T. Kagawa, and K. Utsumiya. “Free hand interface for controlling applications based on Wii remote IR sensor.” In Proc. of the 9th ACM SIGGRAPH Conf. on Virtual-Reality Continuum and its Applications in Industry (VRCAI '10). ACM, New York, NY, USA, 139-142.

[5] D. Natapov, S.J. Castellucci, and I.S. MacKenzie. “ISO 9241-9 evaluation of video game controllers.” In Proc. of Graphics Interface 2009 (GI '09). Canadian Information Processing Society, Toronto, Ont., Canada, Canada, 223-230.

[6] I. Oikonomidis, N. Kyriazis, and A. Argyros. “Efficient model-based 3d tracking of hand articulations using kinect.” Procs. of BMVC, Dundee, UK, pages 1–11, 2011.

[7] L. Rainie. “Tablet and E-book reader Ownership Nearly Double Over the Holiday Gift-Giving Period” Pew Internet. 2012.

[8] G. Shoemaker, L. Findlater, J.Q. Dawson, and K.S. Booth. “Mid-air text input techniques for very large wall displays” In Proc. of Graphics Interface 2009 (GI '09). pp. 231-238.

[9] I.S. MacKenzie and S.X. Zhang. “The immediate usability of graffiti.” In Proc. of the conf. on Graphics interface '97, Toronto, Ont., Canada, Canada, 129-137.

[10] D.C. McCallum and P. Irani. “ARC-Pad: absolute+relative cursor positioning for large displays with a mobile touchscreen.” In Proc. of the 22nd annual ACM symposium on User interface software and technology (UIST '09). ACM, New York, NY, USA, 153-156.

[11] T. Ni, D. Bowman, and C. North. “AirStroke: bringing unistroke text entry to freehand gesture interfaces.” In Proc. of the 2011 annual conference on Human factors in computing systems (CHI '11). ACM, New York, NY, USA, 2473-2476.

[12] YouTube, 2012. YouTube Press Statistics: http://www.youtube.com/t/press_statistics

[13] Netflix, “Netflix Q4 2011 Letter to Shareholders.” http://goo.gl/4vgP1

[14] A. D. Wilson and M. Agrawala. “Text entry using a dual joystick game controller.” In Proc. of the SIGCHI conf. on Human Factors in computing systems (CHI '06), ACM, New York, NY, USA, 475-478.

[15] T. Költringer, P. Isokoski, and T. Grechenig. “TwoStick: writing with a game controller.” In Proc. of Graphics Interface 2007 (GI '07). ACM, New York, NY, USA, 103-110.

[16] Nielsen. “State of the Media: Consumer Usage Report,” 2011. http://goo.gl/PzAzc

[17] Sandvine, 2011. “Sandvine. Global Internet Phenomena Spotlight: Netflix Rising.” http://goo.gl/dbkPz

[18] Hulu, 2012. “Hulu Financial Results:” http://blog.hulu.com/2012/01/12/2011-2012-and-beyond/

[19] Nielsen, 2011. “Nielsen: The Cross-Platform Report.” http://goo.gl/JxEpn

[20] A. Oulasvirta and J. Bergstrom-Lehtovirta. “Ease of juggling: studying the effects of manual multitasking.” In Proc. of the 2011 annual conf. on Human factors in computing systems (CHI '11). ACM, New York, NY, USA, 3103-3112.

[21] NTIA, 2010. “Exploring the Digital Nation: Home Broadband Adoption in the United States.” http://goo.gl/ErsnS

[22] A. Pavlovych & W. Stuerzlinger. “The tradeoff between spatial jitter and latency in pointing tasks.” In Proc. of the 1st ACM SIGCHI symposium on Engineering interactive computing systems (EICS '09). ACM, New York, NY, USA, 187-196.

22. Conclusion

The first manuscript highlights the importance of creating integrated development environments that cater to the motivational needs of both traditional developers and end-user programmers. Additionally we see that end-users are typically not considered when designing more function-based features.

The second manuscript depicts a common occurrence when end-users experience a new application platform such as interactive television. User experience is often lacking which can inspire a consumer to find homespun versions that meet their needs. Just being exposed to the new technology can also inspire end-users to create software solutions in their everyday life.

With the exploding ubiquitous computing industry and the large-scale personal use of multiple devices, we expect to see the total number of end-user app programmers to dramatically increase in the near future. That said, it becomes increasingly important to provide tools that meet the motivational and functional needs of these potential programmers.

23. Bibliography

[1] Five Star Equities, Press release Fri, Oct 19, 2012 8:20 AM EDT http://finance.yahoo.com/news/number-smartphones-around-world-top-122000896.html

[2] Deci, E. L., & Ryan, R. M. (1985). Self-Determination. John Wiley & Sons, Inc.

[3] Ryan, R. M., & Deci, E. L. (2000). Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being. American psychologist, 55(1), 68-78.

24. Appendix

Table 24.1 Summary of motivators enhancing basic psychological needs in software engineering literature [3]

ID	Software Engineering Motivators [3]	Basic Psychological Need	Motivation Type	# of articles studies supporting [3]
SEE1	Career Path (opportunity for advancement, promotion prospect, career planning)	Autonomy	Extrinsic	15
SEE2	Empowerment/responsibility (where responsibility is assigned to the person not the task)	Autonomy	Intrinsic	6
SEE3	Work/life balance (flexibility in work times, caring manager/employer, work location)	Autonomy	Intrinsic	7
SEE4	Autonomy (e.g. freedom to carry out tasks, allowing roles to evolve)	Autonomy	Intrinsic	9
SEE5	Development needs addressed (e.g. training opportunities to widen skills; opportunity to specialize)	Autonomy	Intrinsic	11
SEE6	Variety of Work (e.g. making good use of skills, being stretched)	Competence	Intrinsic	14
SEE7	Feedback	Competence	Intrinsic	10
SEE8	Recognition (for a high quality, good job done based on objective criteria -different from making sure that there are rewards available).	Competence	Intrinsic	12
SEE9	Technically challenging work	Competence	Intrinsic	11
SEE10	Appropriate working conditions/environment/good equipment/tools/physical space/quiet	Competence	Intrinsic	6
SEE11	Sufficient resources	Competence	Intrinsic	2
SEE12	Good management (senior management support,	Relatedness	Intrinsic	16

	team- building, good communication)			
SEE13	Sense of belonging/supportive relationships	Relatedness	Intrinsic	14
SEE14	Employee participation/involvement/working with others	Relatedness	Intrinsic	16
SEE15	Equity	Relatedness	Intrinsic	3
SEE16	Trust/respect	Relatedness	Intrinsic	4
SEE17	Identify with the task (clear goals, personal interest, know purpose of task, how it fits in with whole, job satisfaction; producing identifiable piece of quality work)	Relatedness	Intrinsic	20
SEE18	Making a contribution/task significance (degree to which the job has a substantial impact on the lives or work of other people)	Relatedness	Intrinsic	6
SEE19	Working in successful company (e.g. financially stable)	-	Extrinsic	2
SEE20	Job security/stable environment	-	Extrinsic	10
SEE21	Rewards and incentives (e.g. scope for increased pay and benefits linked to performance)	-	Extrinsic	14

Table 24.2 Summary of de-motivators enhancing basic psychological needs in software engineering literature [3] (SEU)

ID	Software Engineering De-Motivators [3]	Basic Psychological Need	Motivation Type	# of articles supporting [3]
SEU1	Poor management (e.g. poorly conducted meetings that are a waste of time)	Autonomy	Intrinsic	7
SEU2	Lack of influence/not involved in decision making/no voice	Autonomy	Intrinsic	2
SEU3	Lack of promotion opportunities/stagnation/career plateau/boring work/poor job fit	Competence	Extrinsic	5
SEU4	Interesting work going to other parties (e.g.	Competence	Intrinsic	1

	outsourcing)			
SEU5	Unrealistic goals/ phony deadlines	Competence	Intrinsic	4
SEU6	Producing poor quality software (no sense of accomplishment)	Competence	Intrinsic	3
SEU7	Poor communication (Feedback deficiency/loss of direct contact with all levels of management)	Competence	Intrinsic	5
SEU8	Inequity (e.g. recognition based on management intuition or personal preference)	Relatedness	Intrinsic	4
SEU9	Bad relationship with users and colleagues	Relatedness	Intrinsic	4
SEU10	Risk	-	Extrinsic	1
SEU11	Unfair reward system (e.g. Management rewarded for organizational performance; company benefits based on company rank not merit)	-	Extrinsic	2
SEU12	Uncompetitive pay/poor pay/unpaid overtime	-	Extrinsic	6
SEU13	Stress	Unclear	Unclear	5
SEU14	Poor working environment (e.g., wrong staffing levels/unstable/insecure/lacking in investment and resources; being physically separated from team)	Unclear	Unclear	9
SEU15	Poor cultural fit/stereotyping/role ambiguity	Unclear	Unclear	3

Table 24.3 Methods of supporting and undermining autonomy based on the psychology literature

Assigned ID	**Methods to enhance autonomy**
SDTE1	Treat poor behavior as a motivational problem to be solved
SDTE2	Address motivational problems with flexible language
SDTE3	Identify the root cause of motivational issues and communicate improvement options
SDTE4	Ascertain and validate negative expressions and resistance
SDTE5	Work collaboratively to solve problems
SDTE6	Provide opportunities for self direction

SDTE7	Encourage initiative
SDTE8	Identify interests, preferences and competences
SDTE9	Encourage expression of preferences, interests and competences
SDTE10	Consider another's perspective
SDTE11	Provide growth opportunities
SDTE12	Tolerate failure and promote it as necessary for optimized learning
	Methods to undermine autonomy
SDTU1	Pressure compliance by employing a prescribed way of thinking, feeling or behaving
SDTU2	Pressure behaviors supporting a prescribed outcome
SDTU3	Rely on outer sources of motivation
SDTU4	Use a pressured, rigid, no nonsense communication style
SDTU5	Neglect explanatory rationale
SDTU6	Discourage self direction
SDTU7	Silence negative expressions and conflict resolution
SDTU8	Ignore or invalidate negative expressions and resistance
SDTU9	Attempt to transform negative expression into something more acceptable

Table 24.4 Methods of supporting and undermining relatedness based on the psychology literature.

Assigned ID	**Methods to enhance relatedness**
SDTE13	Provide opportunities for social interaction
SDTE14	Express affection and liking
SDTE15	Voice care for others well-being
SDTE16	Share personal resources such as time, attention and energy
SDTE17	Ensure trust and understanding
SDTE18	Encourage obligational support in exchange relationships
SDTE19	Foster emotionally positive interactions
SDTE20	Support intimate, high quality relationships
SDTE21	Supply clear convincing rationale
SDTE13	Provide opportunities for social interaction
SDTE14	Express affection and liking
SDTE15	Voice care for others well-being

	Methods to undermine relatedness
SDTU10	Diminish opportunity for social interaction
SDTU11	Refrain from expressing affection, fondness or concern for others well-being
SDTU12	Horde resources and advertise their scarcity
SDTU13	Discourage intimate, high quality relationships
SDTU14	Neglect to provide rationale
SDTU15	Foster emotionally negative interactions
SDTU16	Diminish trusting behavior and actions
SDTU17	Encourage entitlement
SDTU10	Diminish opportunity for social interaction

Table 24.5 Methods of supporting and undermining competence based on the psychology literature.

Assigned ID	**Methods to enhance competence**
SDTE22	Provide structure
SDTE23	Communicate clearly
SDTE24	Ensure optimal challenge
SDTE25	Enhance flow
SDTE26	Provide information rich guidance and feedback
SDTE27	Tolerate failure and present it as optimal for optimized learning
	Methods to undermine competence
SDTU18	Ensure tasks require a more advanced skill than retained
SDTU19	Provide over simplistic tasks
SDTU20	Supply tasks requiring too much or unnecessary work
SDTU21	Provide negative, incomplete or non-existent feedback
SDTU22	Intolerance for failure
SDTU23	Reduce flow
SDTU24	Use nondescript language

Table 24.6 Methods of supporting intrinsic motivation in an IDE based on combined validated methods in motivational CS and SDT literature

Autonomy	Relatedness	Competence	# of supporting methods	Method IDs supporting	Methods to enhance intrinsic motivation in an IDE
✓	x	✓	6	SDTE1, SDTE2, SDTE3, SEE4, SEE5, SDTE26	Use performance metrics to Identify developer amotivation, determine lacking psychological need and remind programmer of relevant supporting features supporting features
✓	✓	✓	8	SDTE4, SDTE5, SDTE7, SDTE9, SDT10, SDTE17, SDTE22, SEE4	Provide a means for developers to express concerns. Paraphrase back concern and ask developer for input
x	✓	✓	5	STDE26, SDTE27, SEE9, STD24, SEE5	Identify lacking developer skills and coach developers to improve while communicating the importance of failing
✓	✓	✓	4	SDTE5, STD13, SEE11, SEE18	Provide a means for developers to work together solving problems and appropriate information to do so
x	✓	x	2	SDTE17, SEE16	Find a way that helps developers understand how their team mates code and work
✓	x	x	5	SDTE6, SEE2, SDTE8, STDE9, SEE4	Ensure developers have the freedom to choose coding style and preferences
x	✓	✓	2	SDTE19, SDTE26	Add humorous components and positive feedback when

					something works
✓	✓	✓	11	SDTE10, SDTE9, SDTE13, SDTE14, SDTE15, SDTE16, SDTE17, SDTE19, SDTE20, SEE14, SEE15,	Provide positively moderated means for developers to interact socially and express preferences, interests and competencies while suppressing entitlement and encouraging equity
✓	✓	✓	1	SDTE21, SEE17, SEE4	If certain methods are forced, have clear rationale available.
x	✓	**x**	1	SDTE18,	Mimic open source in exchange relationships and obligational support
x	**x**	✓	1	SDTE24	Customize the developer experience based on skill set and experience
✓	✓	✓	6	SDTE22, SDTE23, SDTE25, SEE10, SDTE24, SEE9	Complete usability studies to ensure clear communication, reduce any unnecessary steps and test intrinsic motivation score, ensure structured work flow, make sure it's in optimal challenge
x	**x**	✓	1	SDTE25	Supply easy methods for getting back into the code produced earlier or by someone else to facilitate a quicker state of flow
✓	**x**	**x**	2	SDTE7, SEE2	Mimic open source environments encouraging some level of initiative and empowerment in tasks
✓	**x**	✓	2	SDT11, SEE1	Endorse skill sets and certifications pertaining to career growth
✓	**x**	**x**	**2**	SEE3, SEE4	Ensure developers can work from multiple locations with appropriate access without being online