Adapting Information Technology Innovations to Country Context

Wangai Njoroge Mambo
Information Technology Department
Jomo Kenyatta University, Kenya
Email: mambown2006 [AT] yahoo.com

Abstract— **Information Technology innovations are sometimes adopted by reading and learning by doing a process that may fail due to innovation not having necessary adoption characteristics or missing some knowledge elements. This can be aided by having consultants guide the process but isn't affordable by many following and catching up firms.**

Adapting a technology by integrating everyday proverbs is proposed as a way increasing adoptability and creation of missing knowledge elements through learning by doing. Analogy between software patterns and proverbs were used to find a way of adapting innovation and formalizing proverbs. Design Thinking XP software development method (DTXP) innovation is selected to demonstrate adaptation

I. INTRODUCTION

Use of information technologies has been increasingly and will continue in near future. Information technology (IT) firms adopt IT innovations to support managing firm and to provide foundations for building their innovations.

IT firms can be categorized as leaders or followers. Leading firms create IT innovations. Most following including catching-up firms don't often have capabilities to create novel technologies.

Follower firms can further be categorized into three categories. Catchup firms improve technology by adapting it during adoption. Innovation is creating novel technology in a market, but in developing countries includes adapting and adopting IT technology new to a firm but not market [1]. The second category firms adopt technology with high level of mastery without adapting it. If firm is a developing country's result is innovation. This is due to lower level of knowledge and capabilites in developing country firms and national innovation system. The last category of follower firms adopt technology with low level of mastery. All categories of follower firms can benefit from IT adaptation for adoption.

There are two senses in which adaptation is used in this paper: improving technology for use and making technology suitable for adoption. The primary objective of paper is adaptation to improve adoptability, a secondary objective is technology improvement.

In IT field no firm innovates all the technologies it needs, making any firm to be a follower in at least some technologies. Therefore adopting technology is an important capability for any firm. For example most firms that develop software don't write their own compilers but select programming languages and compilers to adopt then acquire and master them

Innovation research in computing disciplines is increasing. Rogers innovation adoption theory is one of theories used in Information Technology and Information Systems adoption practice and research. According to the theory characteristics that favor innovation adoption are relative advantage, compatibility, complexity, observability and trialability [2]. These characteristics can be used by a firm in two ways, first as part of criteria for selecting technology to adopt. Second as part of criteria for designing and creating IT innovations. Management software quality attributes are used as part of criteria for selecting software architectures and in similar way adoption characteristics should considered a part of innovation requirements.

Design Thinking Extreme programming (DTXP) software development method was developed by integrating XP and Design Thinking (DT) with aim of making XP innovative [3]. DTXP is selected as IT innovation to demonstrate integration with proverbs to increase adoptability. DTXP has relative advantage over XP as it supports innovation. DTXP is compatible with most XP values as DT is an addition. One with experience of XP or DT can try DTXP on limited basis. The complexity of DTXP is comparable to that of XP.

Adapting a technology by integrating proverbs is proposed as a way of increasing adoptability and to support creation missing knowledge elements. The IT technology selected must already be adoptabe by firms in industry. This type of adaptation aims help with technology tranfer to a different country context. DTXP was selected for four reasons, first it's a modular and architectural innovation of XP. The second is it uses DT to catalyze innovation. This is complemented by design oriented approaches increasingly being used to increase innovation in technologies, industries, disciplines and countries, this trend is likely to favor more application of DT in IT. Fourthly DTXP has desired adoption characteristics due to XP and DT being familiar to some IT specialists. Integration with familiar proverbs increases its adoptability in new contexts. Analogy between software patterns and Proverbs is used to find way of incorporating proverbs in DTXP

Architectural innovation introduces new relationships between technology modules, while modular innovation introduces new knowledge in modules [4]. DTXP introduces new relationships between XP method elements and new knowledge DT elements.

Local languages can be used to adapt technologies to country context [5]. Proverbs are specialized part of language and was first reason they were selected for exploration, latter during the research they were found to be similar to software patterns and finally it was discovered some software patterns were proverbs.

The goal of this study is to find ways of adapting the technology innovations to country context. The questions the study aims to answer are:

1. How can proverbs be incorporated in IT innovations to make them more adoptable
2. How can proverbs be formalized to be suitable for use in IT development.

In following sections review of knowledge of DT and XP and their integration in DTXP is carried out. Proverbs are then used to represent this knowledge in same way that software patterns do.

II. XP METHOD

XP and Scrum are most widely used agile methods. Agile methods were developed in response to limitations of plan based methods that are heavy weight which when applied to small and medium projects resulted in excesses in some development activities. Agile methods response to these excesses was to find point where increasing returns ended and diminishing returns set in software development (SD) tasks and end the task just then.

The principles of agile methods [6] are:

1. Customer involved in entire SD process
2. Software developed in small customer specified increments.
3. People not process, the skills and knowledge should drive software development and not other way round
4. Embrace change, Changes are guaranteed in SD, instead of avoiding them design for change
5. Maintain simplicity, parsimony used to select the simplest way that gets work done to acceptable level

Individual mastery of agile methods is a staged process with three stages: follow, detach and fluent [7]. A follower needs to learn one method that works and be able to use it. This way the follower gains experience. After mastering one method and learning its limitations follower is ready to learn another method and move to detaching level. The detacher learns other methods and uses them when appropriate to develop systems.

A fluent developer can follow a method, improvise one or use fragments from different mastered methods [7]. Developer visualizes and strategizes a good way of achieving good results then realizes that strategy, using a method, a combination of different methods elements or no method.

The three levels have a higher level missing, the method creator. The method creator invents, innovates or extends a method, by research, improvisation or innovation. A reliable example is research about method improvement or creation published in reputable peer reviewed journals. The creators of various agile methods such as XP, DTXP, Scrum, Crystal have attained this level.

III. DESIGN THINKING

DT uses design knowledge in thinking how to solve technical and non technical problems. It originated in design but has moved to computing, management and other disciplines.

By a designer visualizing his or her thoughts about aspects of project designer expands the problem space of task, to the extent of including or discovering new aspects [8]. The designer or team integrates and pulls together many different inputs, inspirations and ideas to focus on design problem [9]. Creativity is catalyzed by diversity of elements. If some of the different elements integrated have never been integrated before, are new or the integration is novel it results in creativity. In a similar way research literature review pulls together and integrates variety of selected elements from existing literature.

Designers use knowledge of others, resulting in emergence of learning strategies consisting of cognitive patterns to grasp multiple knowledge and multiple perspectives of others for synthesizing and creative transformation knowledge into products and services [10]. These cognitive patterns have enabled DT and Design science research (DSR), methods based on design to move into other disciplines. DSR is a research method that synthesizes multiple knowledge and perspectives of different people, organizations and disciples, making it a transdisciplinary method. It's already being used in computing and may become one of its major research methods.

With diffusion of mobile technologies to all members of society, in all countries an IT stakeholder can include anyone, creating need for incorporating everyday elements in IT development methods. Integrating DT and agile methods extends agile problem solving paradigm. This is paradigm stretching [11] that extends the boundaries of agile method into innovation field. Integrating DTXP and proverbs extends DTXP boundaries to everyday life.

A characteristic of DT solutions is overlapping problem and solution development that is shared by an XP iteration which overlaps analysis, design and implementation steps.

Agile SD thinking tends to avoid divergent thinking in order to maintain overall view on what to do next [10]. A narrow

and short term focus limits creative thinking and introducing DT in agile IT projects aims to remedy this.

The forces that shape DT are national, disciplinary, corporate cultures and social institutional environment [12] as shown in Figure 1. The national culture includes everyday thinking constructs like proverbs. Proverbs are also sometimes used in disciplinary and corporate culture as seen in their publications.

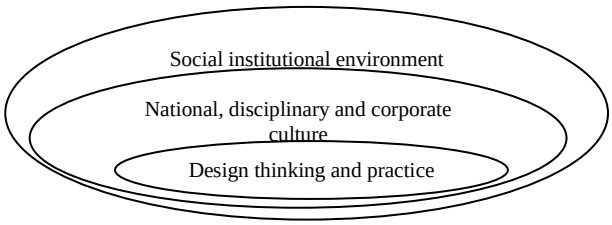

Figure 1 A nested view of design thinking and practice [12]

The following section takes identified DTXP innovation knowledge elements from reviewed and extended literature explores way of representing them with proverbs.

IV. PROVERBS AS KNOWLEDGE REPRESENTATION LANGUAGE

Analogy is a way of understanding something new in terms of something familiar with a known solution. Proverbs are similar to software patterns. This similarity was basis for finding whether patterns solutions can be used to derive adaptation solutions for proverbs. Similarity between DT and XP was basis for integrating them to produce DTXP [3], a way of improving a solution by integrating selected complementary elements of one method into another.

Proverbs like patterns can be used as a way of representing problem solving knowledge. This study views a proverb as belonging to a language if it's originated in that language or it's widely used by speakers of language.

Multiple languages are used by software development teams in some countries and global companies. To illustrate this Sub Saharan Africa is used. The official languages used for business and professional practice are English, French and Portuguese. There are also national and local languages. Most organizations ignore presence African languages which are used by team members to communicate and collaborate. An exception is a Zimbabwean agile software developing company [5]:

> The company uses English as official language and in its working culture uses both English and local languages. This has been used as an element for adapting agile development methods to country context.

When software developers use local languages while developing software, it could eventually become part of corporate culture. Using local languages is one way of adapting technology but it leads to developing islands of expertise, because of barriers of translating between local languages and official languages. Islands of expertise violate goal of some technologies such XP and IT Knowledge management of having project expertise uniformly shared within the team. Proverbs as an alternative to languages have advantage of being easily translated once from one language to another.

Proverbs encapsulate local altitudes and insights [13] thus when used to represent knowledge elements they adapt the innovation to country context by including the local attitudes and insights. To show how proverbs can be used, three African proverbs from [13] and two other widely used proverbs are selected. "If nothing has been forged where has the charcoal gone" this emphasizes effectiveness a core concept of XP method that measures success by amount of software produced. Lean agile method has a principle that incomplete work or activities are waste [14] if they cannot be completed an economical way. "The hidden serpent grows large" also "A stitch in time saves nine" are two proverbs emphasizing importance of early discovery of errors as it's cheaper to correct them. A goal of SD and XP methods is to discover and correct errors as soon as possible

Design patterns are general solutions to repeating and commonly occurring problems in software design that are reused reducing costs and risks associated with uncertainty [15]. They are micro solutions that simplify development of software systems. This definition is extended to software patterns by substituting software design with software activity.

Some English proverbs are used as software patterns like "Divide and conquer" and "Chicken and egg". Divide and conquer is used to divide a large difficult to manage organization into sub organizations [16], it's a problem solving strategy for solving complex problems, a principle that guides software engineering requirements, analysis, modeling and design[17]. It's science dominant strategy reductionism, also used in all disciplines. It is thus transdiciplinary. Chicken and egg pattern is used in teaching interdependent SD concepts concurrently in cases you cannot understand one concept completely before understanding partially another dependent concept [18]. This pattern applies to complex IT adoption in that theory assimilation cannot be completely mastered without technology application and vice versa. The two patterns should be viewed from two perspectives as patterns and as proverbs. A proverb like a software pattern is a solution to a frequently occurring problem but in a different domain of everyday life.

Somerville agile principles can be justified and supported with proverbs. Customer involvement in software development process enables knowledge sharing. This is captured by proverb "one who doesn't know this knows that". Any person with something to contribute should be included within constraints of project.

Principles of developing software in small increments and by incremental development result in principle of maintaining simplicity and are represented by "Divide and conquer" proverb pattern. Change is only constant in technology development. IT development aims to change the world, and its foundations and environment are also continuously changing. Relationship between IT development and change are "chicken and egg" you can't be effective by focusing on one and ignoring the other.

Proverbs are polysmbolic [19] they contains variable symbol spots where other concepts or terms can be substituted. The variable spots are one of factors enabling proverbs to be used in wide variety of situations. Polysymbolism can be used to represent IT common and novel knowledge. For example the proverb "A proverb is horse that can carry you swiftly to discover new ideas". The variable spots are proverb, horse, discover and ideas. Table 1 indicates concepts that can be substituted in their place. The first row represents variable symbol spots and the concepts in columns of other rows concepts that can be substituted. Zero or more variable spots can be substituted and not all combination of substitutable concepts is applicable.

proverb	**horse**	**discover**	**ideas**
knowledge	vehicle	innovate	technology
technique	technology	invent	technique
tool	motivator	improvise	knowledge
thinking	map	develop	tool

Table 1: substitutable concepts in horse proverb

Polysymbolism also applies to patterns and can be used to extend them in a new direction such as innovative direction. Some pattern theories can be used to improve proverbs, while some proverb theories like polysybolism can be used to extend patterns. Analogy is commonly applied in one direction from familiar to unfamiliar. Application in reverse direction is a possibility in some cases

Proverbs can be formalized in pattern format. Some pattern representations use the Alexander format as adapted to software by [20]. This format is used with an additional forces element from [17]. The pattern format used is:

Pattern Name: gives idea of solution and provides guidance on how to achieve the solution. A developer familiar with proverb be assisted by everyday proverb application.
Problem: the problem the pattern solves
Forces: system of forces that affect solving the problem, the constraints and limitations
Solution: - description of resulting solution after applying the pattern
Consequences: The tradeoff of applying the pattern

Three proverbs are represented in pattern format:
Proverb Name: A proverb is horse that can carry you swiftly to discover new ideas.
Problem: discovery, invention, innovation and development of inputs and ideas needed for technology adoption
Forces: having right skills, selecting the right techniques and using them right within constraints of project is critical in developing a solution
Solution: - in technology adoption missing knowledge elements need to be developed, discovered, invented or innovated
Consequences: extending problem solving in new directions increases risk of failure. Results in discovery of new knowledge or it it's not feasible. Failure may be an opportunity to learn

Proverb name: If nothing has been forged where has the charcoal gone?
***Problem*:** outcome not proportional to efforts
Forces: The current level of knowledge and skills determines how problems are solved. Efforts must be focused on solving the real critical problems. Innovation and creativity should be used for improvement.
Solution: Resulting solution should add value to both customers and developers.
Consequences: software produced should be proportional to efforts and resources

Proverb name: The hidden serpent grows larger
Problem: failure to correct errors early
Forces: finding and correcting takes time, this adds to costs
Solution: focus using approaches and techniques that reduce number of errors made and correct any errors made promptly. The longer it takes to find an error and correct it the more expensive it is.
Consequences: The fewer and earlier errors are found the lower the cost of development

V. INTEGRATED DESIGN THINKING AND XP METHOD

DTXP is one of methods developed by integrating design thinking with incremental and iterative software development by incorporating DT elements at the beginning of every iteration like divergent thinking followed by convergent thinking [3]. It's one of few but increasing innovative software methods.

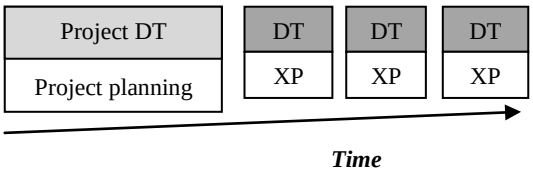

Figure 2. DTXP software development [3] with project DT and project planning step added

DTXP software development consists of project design thinking, project planning, then iteration DT and iteration development as follows:

Project Design thinking: Project creative thinking
Project planning – Minimal planning of SD with awareness that changes will be made to plan.
Iteration Design thinking: Creative thinking about iteration to be built
Iteration planning: The iteration to built is planned
Analysis – User stories represent requirements for iteration to be built
Iteration Design: the iteration is designed
Iteration Implementation: the iteration is implemented using pair programming

VI. ADAPTING DTXP METHOD TO COUNTRY CONTEXT

Adapting DTXP method to country context is done by representing knowledge and experience with proverbs that is then formalized in pattern format. Any changes or new additions to identified proverb patterns can be suggested by team members at end of project

Adaptation process followed is technology independent and thus applicable to all IT innovations. The following steps used to adapt DTXP innovation to country context can be applied to other technology innovations:

1. Carry out a literature review of novel, non novel IT innovation components and integrated whole. The innovation components literature review is likely to result in new understanding and optionally innovation.
2. Search for proverbs that can be used to represent some of the resulting IT knowledge elements found in literature. If found formalize in pattern format the not yet formalized proverbs
3. Apply the adapted IT innovation and evaluate if the adaptation is useful
4. Retain useful proverbs and discard those that are not

VII. CONCLUSIONS

Proverbs have been demonstrated as a candidate technique for adapting IT innovations to country context using DTXP method example. Four proverbs were used to show how to adapt innovations, three of which were formalized in pattern format.

Proverbs use as software patterns were found to have been used in pattern literature. The use of proverbs as way of increasing of adoptability of IT innovations is new contribution of this work. The familiarity of proverbs and selecting ones compatible with IT innovation values is argued will promote adoption and learning by doing. Proverbs use is society wide that makes it easier to include any member of society as stake holder in IT development. This makes IT technology less alien to some stakeholders who until recently were excluded for IT technologies by the digital divide.

Research to compare proverbs and local languages use in technology adaptation should be done to determine their strengths and weaknesses. Famous saying are similar proverbs and software patterns, research should be done to determine their suitability for representing IT innovation knowledge

REFERENCES

[1] Kesidou E. Local knowledge spillovers in high tech clusters in developing countries: The case of Uruguayan software cluster, PhD thesis, University of Eindhoven, 2007.

[2] Rogers E. M. Diffusion of innovations, Free press, 1995.

[3] Hirschfield R. Steinert B. Lincke J. "Agile software development in virtual collaboration environments", In Design thinking: Understand improve apply, Meinel C. and Leifer L. , Eds,. Springer, 2011, pp. 197-218.

[4] Henderson R. M. and Clark K. B. "Architectural innovation: The reconfiguration of existing product technology and failure of established firms", Administrative science quarterly, vol. 35, 1990, pp. 9-30.

[5] Mnkandla E. A selection framework for agile methodology practices: A family of methodologies approach, PhD thesis, university of Witwatersrand, 2008.

[6] Somerville I. (2014) Software engineering, Dorling Kindersley, Pearson Education ninth edition.

[7] Cockburn A. Agile software development: The cooperative game, Pearson Education Inc., 2007.

[8] Tschimmel K. "Design thinking an effective Toolkit for innovation" in proceedings of XXIII ISPIM Conference: Action for innovation: Innovating from experience, Barcelona, pp. 1-21, 2012.

[9] Hartson R. Pyla P. The UX book, Morgan Kaufmann publishers, 2012.

[10] Lindberg T. Meinel C. Wagner R. "Design thinking a useful concept for IT development?" in Design thinking: understand improve apply, Meinel C. and Leifer L., Eds, Springer, 2011, pp. 3-18.

[11] McFadzean E. "Enhancing creative thinking in organizations", Management decision vol. 36 no. 5, 1998, pp. 309-315.

[12] Hinds P. Lyon J. "Innovation and culture: Exploring the work of designers across the globe" in Design thinking: understand, improve apply, Meinel C. and Leifer L., Eds, Springer, 2011, pp. 101-110.

[13] Easton supervised and local researchers, "Participatory management and local culture: Proverbs and paradigms". In Indigenous knowledge: Local paths to Global development, Knowledge and learning group, World Bank, 2004, pp. 128-131.

[14] Misaghi M. Bosnic J. "Lean mindset in software engineering : A case study of software house in Brazilian state of Catarina", In Knowledge based Software engineering, Kravets et al. , Eds, 2011, Springer ??

[15] Joosen W. Lopez J. Martinelli F. Massacci F. "Engineering secure future internet services", In future of Internet, Domingue et al, Eds, Springer 2011, pp. 177-191.

[16] Coplien J. "A generative development: process pattern language", In Pattern languages of program design, Coplien and Schmidt, Eds, Addison-Wesley Company, 1995, pp. 391-406.

[17] Pressman R. Software Engineering: A practitioners Approach, McGraw-Hill Company, 2010.

[18] D. L. G. Anthony "Patterns for classroom education", in Pattern languages of program design, vol 2, J. Coplien, N. Kerth, Addison-Wesley Company, 1995, pp. 391-406.

[19] Wanjohi G. Philosophy and wisdom of African proverbs, Pauline publications, 1997.

[20] Gamma E. Helm R. Johnson R. Vlissides J. Design patterns: Elements of reusable object oriented software, Addison-Wesley, 1995.