Cognitive Complexity and Methodical Training: Enhancing or Suppressing Creativity

Jungwoo Lee, Ph.D.
Assistant Professor of MIS
College of Business
UNLV
4505 Maryland Parkway
Las Vegas, NV 89154-6009
jungwoo@acm.org

Duane P. Truex, Ph.D.
Assistant Professor of CIS
J. Mack Robinson College of Business Admin
Georgia State University
35 Broad Street
Atlanta, GA 30302-4015
dtruex@gsu.edu

Abstract

This research explores an aspect of developers’ cognitive structure based on the theory of cognitive complexity. Underlying structure for information processing is the central topic of a variety of theories known collectively as complexity theories. These theories address the structural dimensions that underlie the flow, processing, and use of information within individual, team or organization. This paper uses the theory of complexity as a theoretical basis in explaining phenomena in systems development arena, especially in relation to training on systems development methods. Based on an experiment of novice developers, this paper provides initial evidence for research feasibility using the theory of cognitive complexity, suggests theoretical relationship of the cognitive complexity towards the creativity, and presents a stages-of-growth model for cognitive development of systems developer.

1. Introduction

Some information systems development projects flourish, while others collapse [2-4]. Researchers have found various reasons why some project fails. Many attribute the failure to procedural and organizational issues. Although systems development projects involve a complex process of management encompassing many factors beyond individual’s control, apparently good part of any systems development is a product of its working group members - the systems developers [5].

Considering the importance of the role the individual developer plays and given the complexity involved in systems development, understanding the individual developer’s cognition is critical in understanding the current phenomenology of systems development. However, it is not well known how individual systems developers are actually approaching the systems development process and we still have many questions not answered in terms of their cognitive content, structure and process. How is information filtered in the developer’s cognition? What are the cognitive structural characteristics of developers that perform well even in the face of adversity? What are the structural characteristics of those that fail, even in a favorable environment? Answers to these questions are of considerable importance.

A variety of theories known collectively as the theory of complexity addresses cognitive structural information processing [6-8]. The complexity theories address the structural dimensions that underlie the flow, processing, and use of information within individual, team or organization [9, p. 2]. This paper uses the theory of complexity as a theoretical basis in exploring one of the basic questions about developer’s cognitive structure, specifically in relation to methodical training on systems development. An experiment is conducted with novice developers and the result provides initial evidence for the utility of the complexity theory in this area.

The remainder of this paper is divided into three major sections: review of theoretical background, description of the research project, and discussion. The discussion section suggests future directions in continuing exploration of systems developers’ cognitive structure.

2. Theoretical Background

2.1. Theory of Cognitive Complexity

Cognitive complexity is an individual characteristic that represents the degree to which the individual uses information to apply multiple perspectives when perceiving and evaluating stimuli. Streufert [9] defines cognitive complexity as follows.
Cognitive Complexity-Simplicity: Represents the degree to which a potentially multidimensional cognitive space is differentiated and integrated. A cognitively complex person would employ differentiation and integration as part of his or her information processing. In other words, that person’s cognitive structure would likely function multidimensionally. A less complex person would respond to stimulus arrays on the basis of few or only one dimension – that is, would demonstrate less, little, or no dimensional differentiation and integration. At the extreme, such a person would function in unidimensional fashion in response to any or all stimuli [9, p. 18].

This definition of cognitive complexity focuses on differentiation and integration. An individual maintains multidimensional cognitive structure and used it when evaluating environmental stimuli. In using this multidimensional structure, the individual functions both as a potential differentiator and integrator across these dimensions. This flexible adaptation within multidimensional structure makes the theory of complexity different from mathematical decision-making approaches in which inflexible hierarchical system of dimensions is assumed fixed. This multidimensionality theorem, on which the theory of cognitive complexity is based, can be traced back to the origin – the Personal Construct Theory [10].

In clinical psychology, Kelly [10] theorizes that an individual’s perception of the world is represented in the form of constructs because a person’s cognitive processes are psychologically channeled by the ways in which one anticipates events. In construing the surrounding world, a person evaluates a series of events. These events are characterized by comparing them to each other. In doing this comparison, a person uses personal constructs that involve similarity and contrast in any construct:

A construct which implied similarity without contrast would represent just as much of a chaotic undifferentiated homogeneity as a construct which implied contrast without similarity would represent a chaotic particularized heterogeneity. The former would leave the person engulfed in a sea with no landmarks to relieve the monotony; the latter would confront him with an interminable series of kaleidoscopic changes in which nothing would ever appear familiar. [10, p. 51]

Theorists after Kelly devoted their time on how to interpret this ‘system of personal constructs’ and developed ideas such as hierarchical superordination and discrimination [11], differentiation and integration [6], and interaction between environmental and internal complexity [12]. The theory of complexity is one of its derivatives. In terms of practical applications, the theory of complexity has been employed in research on communication, information orientation of individual, attitude and creativity. A cognitively complex individual is more effective in tasks dependent on communication [13], also more resistant to persuasive attacks [14], more effective in terms of information utilization [15], spreads perceptual cognitive categories more evenly [16], is more tolerant for inconsistent verbal messages [17], and more creative [18, 19].

Researchers also have noted the relationship between cognitive complexity and leadership behaviors. Groups headed by cognitively complex leaders perform better than groups led by noncomplex leaders [20]. The organizational change literature indicates that managers must establish organizations that adapt and learn [21] in which the organizational leader’s role is to sustain an environment that promotes the ability to cope and creatively view the environmental cues. Cockrill [22] suggests that high performing managers in dynamic environments are able to identify multiple options – cognitively complex - in planning and decision-making and are capable of focusing on and evaluating different options simultaneously. It can be concluded that cognitively complex individuals can look at situations from a number of different viewpoints (using integration and differentiation across these viewpoints), whereas simple individuals use few viewpoints.

Recent complexity theory focuses on the fit between environmental conditions and degrees of cognitive complexity, recognizing inconsistent research outcome in which cognitively complex persons do not perform well in all conditions. They found that cognitively complex individual performs well when their job contains high complexity. In other words, when environmental

![Diagram](image-url)  
**Figure 1.** The Relationships among cognitive complexity, environmental complexity and task behavior, adopted from [9] p. 26
complexity is low, low complexity persons may perform better as can be seen in Figure 1.

2.2 Complexity Involved In Developing Information Systems

Without question, information systems development is a complex task. Perhaps an individual programmer’s task is simple, but front-end analysis and design of an information system for a specific situation is not a simple task. The difficulties inherent in systems development have been documented in the information systems literature. Langefors [23] first describes this difficulty as “imperceivability.” Due to the inherent limitations of human cognitive capabilities, systems developers cannot perceive all aspects of the complex real world situation and thus perceive only limited aspects of the problem situation. Brooks [5] even suggests that the essential problems in information systems development may not be addressed by development methods, but by creative individuals.

The complexity of software is an essential property, not an accidental one. Hence, descriptions of a software entity that abstracts away its complexity often abstracts away its essences... This paradigm [the limited perspective, complex modeling] worked [in mathematics and physical sciences] because the complexities ignored in the models were not the essential properties of the phenomena [in these fields]. It does not work when the complexities are the essence [such as the case of information systems development].

Interestingly, researchers are finding that the actual process of systems development does not strictly follow the development methods that have been formalized within firms or articulated in public forums [24-26]. The reason for the actual process of development being very distant from what has been described in formal textbooks can be inferred by extending Brooks’ prior argument: the actual practices of systems development may contain many factors or components which possibly cannot be addressed by any current development method. Aspects of perspectives of the situation covered by development methods are well-structured, while the unaddressed parts are primarily the ill-structured portion of the situation, which may only be addressed by developer’s creativity[27].

The field of systems development is experiencing a serious theoretical argument for and against so-called ‘development methods.’ It remains unclear till now whether the development process is a codifiable methodical task or a craftsmanship-like task that involves creative individuals. This argument for or against methodical rigor in developing information system can be interpreted as an argument for or against the role of developer’s creativity in the complex environment of systems development.

2.3 Cognitive complexity and Creativity

Creativity in business and organizational settings implies more than just using a specific technique. It involves producing patented products, organizing innovative services, and establishing new organizational procedures. It implies the ability to view interrelationships among seemingly unrelated components in a coherent pattern. Jackson and Messick [28] argues that creativity is not just unusual or a remote association but also involves transformation. Streufert and Streufert [6] extend the view that this transformation is a structural characteristic, reflecting some level of cognitive activity of integration.

Integration: The process of relating a stimulus configuration of two or more orthogonal or oblique dimensions, systems, or subsystems in cognitive or conceptual space to produce an outcome that is determined by the joint demands of each dimension, system, or subsystem involved [6, p. 17].

A high-level integrator has the potential for creative thought and action. A theoretical link between cognitive complexity and creativity can be established here: a cognitive complex integrator should have the potential for creative thoughts and action depending upon other mediating variables such as intelligence, knowledge, technical skills, special talents, politico-religious factors, cultural factors, socioeconomic factors, educational factors, internal motivation, confidence, non-conformity.

[29]

The literature on cognitive complexity suggests developers’ cognitive complexity level changes before and after the training on a set of development methods. It is expected that the rigorous methodical training decrease the level of cognitive complexity because this kind of training forces people to think in terms of predefined procedures and artifacts. From the literature, it can also be inferred that reduced cognitive complexity leads to reduced creativity. Taking the view that the cognitive complexity is malleable through training and experience rather than taking it as unchangeable personality trait, an instrument is developed here and administered to measure the changes in cognitive complexity before and after a methodical training. The following sections report on research design and outcomes. Discussion of results follows.
3. Research Design

The primary treatment in this research is the training on structured methods, which is hypothesized to influence the formation of cognitive structures – especially cognitive complexity. The target of opportunity exists only in a university setting where the precedence of prerequisites in curriculum prevents students from taking systems development methods classes before completing several programming classes and other prerequisite classes. Therefore, student subjects can be naturally grouped into control and treatment groups: students who have taken systems development course (methodical treatment group) and who have not taken systems development course (anti-methodical control group). Therefore, the main study is designed as a survey of student groups using a standard repertory grid style instrument, but for validation, a pilot test was conducted in a laboratory setting.

3.1 Standard Domain-Specific Rep Grid Development

A typical repertory grid consists of three components: (1) elements, which define the area of focus; (2) constructs, which are criteria that subjects are using in differentiating between elements; and (3) a connection mechanism between elements and constructs which can show how each element is being assessed on each construct (See Figure 2 for a sample of the role repertory grid).

Elements define the domain upon which the grid will be based. A set of elements is chosen to represent the area of interest while a construct is, in general, elicited as representing “a way in which two or more things are alike and thereby different from a third or more things” [30, p. 5] in this domain of interest. Elements are listed along the top of a grid, while the vertical axis of a grid is reserved for listing elicited constructs. Within the cells of a grid lies how each element is being assessed on each construct.

Concerning constructs, Kelly provides six assumptions that underlie his original Role Construct Repertory Test. These six principles are equally applicable to construct elicitation in general [10, p. 229, 30, p.14]: (1) the constructs elicited should be permeable, (2) the pre-existing construct should be elicited, (3) the verbal labels attached to the constructs should be communicable, (4) the constructs should represent the subject’s understanding, right or wrong, of the way other people look at things, (5) the subject should not dissociate himself entirely from the elements or from the constructs elicited, and (6) the constructs elicited should be explicitly bipolar.

As this research focuses on structural differences in cognition of systems developers, hypothesized to be caused by the methodical training, the context under which each developer’s construct system operates is different development approaches: methodical versus anti-methodical. Differences in evaluating environmental cues based on different assumptions of methodical and anti-methodical approaches is the domain of this research. In this regard, a set of constructs were developed and derived from the theoretical debate of methodical versus anti-methodical approaches to systems development and shown in Table 1.

3.2 Technical Validation

The technical validation for this instrument consists of

<table>
<thead>
<tr>
<th>Elements</th>
<th>Constructs (1 through 7 points rating scale)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Myself</td>
<td>1 practical</td>
</tr>
<tr>
<td>Training advisor</td>
<td>2 sensitive</td>
</tr>
<tr>
<td>Progressive manager</td>
<td>3 self motivated</td>
</tr>
<tr>
<td>Conservative manager</td>
<td>4 introvert</td>
</tr>
<tr>
<td>An effective trainer</td>
<td>5 committed to holding companies</td>
</tr>
<tr>
<td>My self at end of training</td>
<td>6 work hard</td>
</tr>
<tr>
<td>Myself when I took this job</td>
<td>7 less serious about the job</td>
</tr>
</tbody>
</table>

Figure 2. A Role Repertory Grid
two phases: (1) qualitative review, and (2) reliability test. Three doctoral students participated in this first phase qualitative review. At the end of this initial review, the proposed modifications were: (1) to present six different types of information systems as elements for evaluation instead of system components, and (2) to reformat the questionnaire into a series of statements for evaluation rather than a single page matrix format. Another panel of experts qualitatively reviewed this revised instrument. This second phase expert panel consisted of another three doctoral students, three practitioners and three professors. They were asked to complete the instrument, and follow-up qualitative interviews were conducted by the researcher. In this qualitative review, content and external validity was stressed by encouraging participants to point out ambiguous statements and suggest modifications. This review session was intended to facilitate qualitative revisions, leading to a newly designed repertory grid be ready for a quantitative pretest. By the final version, the instrument had undergone a dramatic change in terms of its content and form.

Reliability. In the context of repertory grid technique, Kelly defined reliability as “characteristic of a test which makes it insensitive to change” [30, p. 82]. Essentially, reliability is a statement about the stability of individual measures across replications from the same source of information (with same subjects in this case). Fransella and Bannister [30] have identified eight different measures of reliability in a repertory grid: mal-distribution, intensity, pattern of construct relationship, specific relationship between constructs, stability of elicited constructs, stability of elements, insight measures, and social dictionary measure. Out of these eight, the last four are specifically applicable to the case when a repertory grid technique is used for idiographic elicitation of constructs and contents. Mal-distribution and specific relationships between constructs are eliminated due to lack of rigor. For this instrument, only the pattern of construct relationships and intensity stability were tested by retesting the same subjects over a period of time.

A section of a systems analysis class was chosen as a pilot test group. At the beginning of a class session, copies of the revised standard repertory grid were handed out. Sixteen students were present at this session. The entire administration lasted about forty minutes including five minutes of instructions at the beginning. One week later, the same instrument was administered again to the same class for a second round. This was necessary to establish test-retest reliability. Only twelve subjects out of

<table>
<thead>
<tr>
<th>Theoretical Nomenclature</th>
<th>Constructs (Methodical versus Anti-Methodical)</th>
<th>Operationalized Statements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epistemology</td>
<td>Objective/Subjective</td>
<td>The internal workings of this system will be the same no matter who build the system.</td>
</tr>
<tr>
<td>Goal Specification</td>
<td>Clear/Fuzzy</td>
<td>The purpose and functions of this system are clear and unambiguous.</td>
</tr>
<tr>
<td>System Scope</td>
<td>Holistic/Localized</td>
<td>To build this system, a clear view of the organizational ‘big picture’ is required.</td>
</tr>
<tr>
<td>Time for Specification</td>
<td>A priori/On-going</td>
<td>The needs, purpose and functionality of this system can be clearly stated even before it is being built.</td>
</tr>
<tr>
<td>System Boundary</td>
<td>Stable/Changing</td>
<td>The needs, purpose, and functionality of this system are likely to remain stable, throughout development process.</td>
</tr>
<tr>
<td>System Objective</td>
<td>Perfect/Working</td>
<td>This system needs to be built perfectly to be of any use. If it is imperfect, the system will not be of any use.</td>
</tr>
<tr>
<td>System Specification</td>
<td>Static/Dynamic</td>
<td>Success in developing this system means the system does exactly what it is initially designed to do.</td>
</tr>
<tr>
<td>Process control</td>
<td>Linear/Evolutionary</td>
<td>Certain steps must be followed exactly to successfully complete this system.</td>
</tr>
<tr>
<td>Metrics</td>
<td>Measurable/Unmeasurable</td>
<td>This system is very much like other systems that I am likely to build</td>
</tr>
<tr>
<td>Ontology</td>
<td>Goal achieving/Survival</td>
<td>Successful completion of this system requires straight-forward commitment to a clear goal, instead of changing courses on the fly.</td>
</tr>
</tbody>
</table>

Table 1. Theoretical Constructs
the first sixteen were present in the retest session. There were five female and seven male students and the average age was 26. They had work experience of two years on average.

Basic analysis of grid data yields a matrix of measures of interrelationship between constructs. For reliability test of the refined repertory grid for this study, reliability indices are calculated following [30]. First the relationship scores of each matrix are rank-ordered in descending order and then Spearman’s rho’s are calculated between the first and second tests. Spearman’s rho of rank-ordered sequence is a non-parametric statistic and it is appropriate here because the normality assumption seems to be violated in our repertory grid for this study, by design. A generally accepted level of reliability coefficient is 0.6 [32]. Spearman’s rho for rank-ordered relationship patterns for constructs and elements was 0.66 for constructs and 0.89 for elements, and both patterns of constructs and elements passed the reliability threshold of 0.6.

4. Data Analysis

4.1 Cognitive Complexity: Intensity Scores

Bannister [33-35] developed the idea of intensity. He argues that there is a relationship between the size of correlation obtained on a rank grid and the notion of tightness-looseness. This operational definition he called the Intensity score. The intensity score is simply the sum of all relationship scores for all constructs. Bieri [11, 36] further developed this idea and defined cognitive complexity as “...the capacity to construe social behaviour in a multidimensional way. A more cognitively complex person has available a more differentiated system of dimensions for perceiving others’ behaviour than does a less cognitively complex individual...” [11. P. 185]

It should be noted here that the Intensity Score presents the cognitive complexity in reverse – low intensity means high complexity and high intensity means cognitively simple.

For this research, intensity scores were calculated using PC Grid program developed by Meltzer [37]. Table 2 presents results of a series of t-tests of intensity scores among groups of IS students in different stages of their academic progression. This result suggests that methodical instructions clearly have an effect of increasing intensity in their thinking, meaning that it is making them cognitively simple, as subjects undergo formal ISD methods training.

This trend is more clearly presented in Figure 3. In this figure note that subject’s intensity scores first decrease as they complete their first visual programming course. But then their intensity scores soar as they experience their first course on systems development.

<table>
<thead>
<tr>
<th></th>
<th>At the end of SAD</th>
<th>At the beginning of SAD</th>
<th>At the end of SAD</th>
<th>At the beginning of SAD</th>
<th>At the end of programming</th>
<th>At the beginning of programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>814.4444</td>
<td>772.5185</td>
<td>772.5185</td>
<td>503.9412</td>
<td>503.9412</td>
<td>554.5000</td>
</tr>
<tr>
<td>Variance</td>
<td>258373.7908</td>
<td>164949.4900</td>
<td>164949.4900</td>
<td>70334.3088</td>
<td>70334.3088</td>
<td>52455.0000</td>
</tr>
<tr>
<td>Observations</td>
<td>18</td>
<td>27</td>
<td>27</td>
<td>17</td>
<td>17</td>
<td>12</td>
</tr>
<tr>
<td>Pooled Variance</td>
<td>201884.6787</td>
<td></td>
<td>128905.6115</td>
<td></td>
<td>63050.1460</td>
<td></td>
</tr>
<tr>
<td>Df</td>
<td>43</td>
<td></td>
<td>42</td>
<td></td>
<td>27</td>
<td></td>
</tr>
<tr>
<td>t Stat</td>
<td>0.3066</td>
<td></td>
<td>2.4161</td>
<td></td>
<td>-0.5340</td>
<td></td>
</tr>
<tr>
<td>P(T&lt;=t)</td>
<td>0.7606</td>
<td></td>
<td>0.0201</td>
<td></td>
<td>0.5977</td>
<td></td>
</tr>
<tr>
<td>t Critical</td>
<td>2.0167</td>
<td></td>
<td>2.0181</td>
<td></td>
<td>2.0518</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. T-tests of Intensity Scores
The results are consistent with a prediction Kelly [10] made that in some contexts the person who could deal with social relationships in relatively differentiated way might be more successful than one who is more tightly constrained in her thinking. This theory of differentiated structure is applicable to the IS development arena in a straightforward manner. The complexity involved in the process and in the environment of systems development, coupled with the diversity of stakeholders, and rapid technological change requires successful developers to remain open and flexible in their thinking.

The essence of Kelly's argument is that we loosen, then tighten, and then loosen our thinking in a cyclic manner. One's aim is first to gain a perspective and have it become concrete. Then after some time, one realizes that this simple and concrete perspective is not sufficient for all situations, and thus gains a new perspective and begins the concretization process again by synthesizing with current perspectives. In doing so, one incrementally reframe one's perspective. Most people undergo this cycle of loosening and tightening.

4.2. Overall Model of Cognitive Development

Extrapolating from this study, one may infer that developers may experience developmental stages with regards to their perception on method, which may relate to their different usage of methods. This is useful in explaining the current methodical versus anti-methodical debate. Instead of looking at these two as antitheses, we can take procedural developmental view. The debate on utility of method can be seen as a reflection of the 'stages of growth' model which most developers experience throughout their career. Authors identified and named following four stage model: Pre-Method, Method, Anti-Method, and Amethodical.

Pre-Method: The first stage of ‘Pre-Method’ refers to a cognitive state of a novice developer (actually a programmer) who has never formally trained on systems development methods. As they do not know anything about methodical approaches their awareness of methods is very low. However, in general, they maintain a positive attitude for methodical approaches because they have experienced difficulties in programming and are interested in learning how to solve programming problems better. They expect methods to be ‘panacea’ for programming problems. This stage refers to the cognitive state of novice programmers who have not been exposed to any formal training on methods. They are simply ignorant of any formalistic methods, except implicit assumptions built into programming languages. They have no idea about the know-how, know-when, and know-why of systems development methods.

Methods: This is when novice programmers are exposed to development methods through formal training, self-learning or the apprenticeship process. In this stage, as they have learned the ‘know-how’ of development methods, they are trying to apply this know-how on any case that comes to them. However, it is a naïve application of their know-how, because they are not aware of the know-when or the know-why of methods that they have learned. There is the expectation that the application of methods will yield a good working system. As they take the first training on methods, they begin to tighten their thinking and maintain the firm and positive position concerning methods. Taking a single perspective learned in their first methods training, they think that this ‘hammer’ is the solution for every problem in systems development. Their attitude becomes more positive and the awareness level increases.

Against Methods: To reach this stage, developers experience the failure of general methods in real systems development projects. At this stage, they have realized the blind application of a method is as dangerous as hammering in a screw. But they are not yet sufficiently versed in the know-when and know-why of different methods. Thus this stage is an expression of the extreme frustration with the constraints and limits of formal ISD methods [38]. As they begin to experience the inherent complexity of systems development through hands-on application building projects such as the ones generally given as course projects in database classes, they become disappointed by limitations of methods. They begin to show symptoms of denial and reject the utility of methods themselves. Henceforth, this stage can be called ‘Against Methods’ in the sense used by [38]. In this stage, their awareness of methods increases slightly but their attitudes become negative towards methodical approaches.

Amethod: At the same time of denial, they also begin loosening their thinking in search of perspectives other than what they have tightly believed, and open up their cognition for other perspectives. Through this search process, successful developers come to realize the know-why and the know-when of different methods. In other words, they now become free from the bias for or against methods. Developers in this stage are the ones who can discern the good side and the bad side of different methods and adapting different method components for specific projects based on situational contingencies and intuition. After gaining multiple perspectives and synthesizing the know-how from several methods and techniques (i.e., several cycles of loosening and tightening of their thinking) including anti-methodical experience, they come to the realization of the know-when and know-why of systems development methods. They have several tools under their belt and use them freely and comfortably. They even know when and why it is appropriate to apply different methods fragments and when one may rely on...
‘intuition.’ This refers to the enlightened use of tools, not subject to the shape of tools or the contingencies.

6. Conclusions and Discussion

Most of the research on information systems development has focused upon developing methods or improving the development process. The field even suffers from a lack of empirical research on the use of development methods. “We don’t know how methodologies are used or how effective they are” [1, p. 186]. As the currently on-going debate for and against the methodical rigor in systems development can be viewed from the perspective of developers’ creativity, it is also important to investigate the relationship between development methods and creativity. This research explored an aspect of developers’ cognitive structure based on the theory of cognitive complexity. It provides a piece of empirical evidence on how development methods are actually being used in terms of its influence on developers’ cognitive structures and presents a developmental theory of systems developers.

This extrapolation concerning developers’ cognitive development needs to be confirmed. This initial research bases its argument on testing novice developers before and after the training. This four-stage growth model needs to be confirmed by extensive and longitudinal field study on experienced systems developers. It will be interesting to see how their cognitive structure changes throughout their career and how their external and internal evaluation of their performance changes. Second phase of this research project will include experienced developers across different culture and the dependent variable of their performance.

References