Complex adaptive systems (CAS) theory characterizes the role of emergence in the world of frequent and continuous change. In the context of information systems (IS) CAS can help conceptualize the phenomenon of IS emergence. In this domain agile development methods were introduced to address speed and the problems of change in IS development. The paper provides a review of agile development practices and their interpretation from a CAS perspective. It is concluded that IS emergence can be realized by engaging development teams through agile practices that are found to support CAS concepts. It is also found that some CAS principles have not been fully realized in current agile methods highlighting possible areas of improvement. Based on this analysis a detailed framework is derived to outline emergence mechanisms in IS development in general. The framework as grounded in CAS theory provides cornerstone elements towards a generic IS emergence theory we refer to as Complex Adaptive Information Systems (CAIS).

INTRODUCTION

The turbulent nature of modern business environments requires organizations to react quickly and creatively to make the most of new opportunities and business models. State-of-the-art IT projects based on service-oriented architecture (SOA) that extends features of client-server architecture, have shown evidence of system and process agility where agile practices have been adopted in design, development and maintenance of such applications (Ren & Lyytinen, 2008; Baskerville et al., 2005). This is because in modern environments new economic realities, such as rapid development of technology and increased global competitiveness make organizations less stable, fast-changing and emergent (Truex et al., 1999). Turbulent environments make businesses unpredictable and hence unplannable in the traditional sense of control and optimization (Riehle, 2000). For this reason agile development...
methods adopt concepts of emergence, such as customer focus, participative design, time-boxed project management, continuous software development and others (Highsmith, 2002). According to VersionOne (2011) that undertook an agile adoption survey; agile practices of common use are daily stand-up meetings, iteration planning, release planning, burn down charts, retrospectives and continuous integration. SCRUM is the most used agile development methodology, adopted among over 50% of survey respondents. SCRUM practices are daily stand-up meetings, sprints that cover release planning, burn down charts to tackle remaining tasks and deviations from schedules, and retrospectives that are feedback, review meetings to assess the sprint to be released and plan for the next one.

Agile development methods were designed to overcome the problems posed by fast-changing environments. Their major benefits include the ability to tackle changing requirements, project visibility and transparency, increased productivity, accelerated time to market and cost reduction (VersionOne, 2011). In spite of such benefits there are still concerns about their suitability in various contexts such as adoption within distributed teams and agile rigor and robustness (Ambler, 2002b; Conboy & Fitzgerald, 2004; VersionOne, 2011). The reason for such concerns is partly because the principles of agile development lack grounding in theory and philosophy.

Complex adaptive systems (CAS) theory specifies emergence mechanisms and characteristics that make a CAS react quickly and creatively to changing necessities. According to Augustine and Woodcock (2003), CAS theory in social and management contexts views organizations as actors with the intrinsic ability to deal with change, interacting with each other and with the business environment. Similarly, IS emergence can be interpreted using complexity concepts. In this paper we apply CAS concepts to agile development and provide interpretation of IS emergence phenomenon.

**COMPLEX ADAPTIVE SYSTEMS (CAS) THEORY**

Complexity science seeks to explain the process of emergence of new properties and the spontaneous creation of order after change. CAS theory originated in the natural sciences and articulates how interacting agents such as organisms adapt and coevolve over time in spontaneous ways (Dooley, 1997). Holland (1995) defines CAS as a system composed of interacting agents, which undergo constant change, both autonomously and in interaction with their environment. He explains that heterogeneous agents exhibit various agent behaviors that can be defined in terms of “simple rules” where they adapt and evolve through their interactions and by changing their rules through learning as experience accumulates. Therefore the
behavior of CAS is typically unpredictable, but yet exhibits various forms of order and regulation (see Figure 1).

Complexity principles emphasize that emergence of properties and creation of new orders are not explicable from a purely reductionist viewpoint, but that the whole is greater than the sum of the parts (Kaufman, 1993). This means the focus of attention shifts from understanding the parts or entities of which the whole was composed to the interaction of subsystems (agents) to form a system. Organizations therefore should not be seen as parts adding to a whole but rather as a corporation in which the interactions between its employees are of primary importance where forms of behavior are determined by the tendency to achieve a certain goal (Stacey et al., 2000).

How do complex non-linear systems with their vast numbers of interacting agents function to produce orderly patterns of behavior? The field of complex adaptive systems seeks rules and principles to interpret this phenomenon. Different authors provided characteristics and/or principles of CAS as summarized in Table 1, along with a brief description on meaning & implications of such principles to emergence in social contexts. According to Middleton-Kelly (2003) many of the identified characteristics are well known in previous theoretical frameworks; for example, the concepts of connectivity, interdependence, feedback and adjustments are familiar from systems dynamics theory. However, complexity theory extends these theories and adds new concepts such as coevolution, self-organization, edge of chaos and historicity and time dependence, which enrich these systems thinking concepts.
**RESEARCH APPROACH**

Agile development methods excel in fast-changing environments with the primary evidence based on practical experiences, but there is little theoretical proof that agile methods facilitate change in emergent environments (Ambler, 2002b; Dybå & Dingsøyr, 2008) Claims of the benefits of the agile approach lie mainly in its focus on building systems in short time periods and being flexible enough to respond to change without descending into chaos (Boehm, 2002). As a result, there are many methods currently in use in the market branded as agile but still they remain vague in their theoretical underpinning and their rigor and reliability.

In order to address this lack of provenance this research aims to interpret agile development by building on complex adaptive systems (CAS) theory with the aim of providing better theoretical grounding for agile development. It also aims to highlight areas in agile development where CAS principles are not fully accommodated.

The paper presents theoretical/conceptual research where the IS emergence phenomenon represented in agile development will be interpreted and conceptualized using complex adaptive system (CAS) theory. Providing theoretical proof to the agile methods that have wide practical acceptance provides triangulation (Yin, 1984) and will hopefully help increase the credibility of agile development.

This paper will tackle theoretical evidence to address the following research questions:

- Can emergent IS development be conceptualized as a complex adaptive system (CAS)?
- Does agile development reflect characteristics of a CAS?
- Are the various principles of CAS fully adopted in agile development?

In order to provide evidence for these propositions, the paper examines agile development principles and practices and then maps them to CAS principles. The research will cover the following steps:

1. Review of CAS principles and their meaning and implications for emergence as shown in the previous section
2. Review of agile development principles and practices
3. Mapping and interpretation of agile principles and practices from CAS perspective
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(P1) Unpredictability</td>
<td>x</td>
<td></td>
<td>x</td>
<td>Unpredictability is the trigger for novel social processes to emerge and force reformations to the reduction of uncertainty (Montuori, 2003).</td>
</tr>
<tr>
<td>(P2) Non-linearity</td>
<td>x</td>
<td></td>
<td></td>
<td>Nonlinearity principle emphasizes that emergence of new order is not explicable from a reductionist viewpoint, but the whole is greater than the sum of the parts (Kaufman, 1993).</td>
</tr>
<tr>
<td>(P3) Variation &amp; diversity</td>
<td>x</td>
<td></td>
<td>x</td>
<td>CASs are made up of heterogeneous agents (Holland, 1995); each agent has different governing rules from the other (Benbya &amp; McKelvey, 2006).</td>
</tr>
<tr>
<td>(P4) Interactions</td>
<td>x</td>
<td></td>
<td></td>
<td>Agents in a CAS undergo constant interactions, both autonomously and with their environment (Heylighen, 2001).</td>
</tr>
<tr>
<td>(P5) Far-from equilibrium/edge of chaos</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>In order to harness change with no anarchy CAS strives to maintain a balance between the completely ordered, “frozen” regime and the completely disordered, chaotic regime, which is known as operating on “edge of chaos” (McKelvey, 1999).</td>
</tr>
<tr>
<td>(P6) Adaptation as fit to environment/context</td>
<td>x</td>
<td>x</td>
<td></td>
<td>Tendency to adapt to a particular situation depends on the context and the chances available in the environment; this will determine the possibilities for change (Keller, 1996).</td>
</tr>
<tr>
<td>(P7) Connectivity &amp; interdependence</td>
<td>x</td>
<td></td>
<td></td>
<td>Structural coupling emphasizes the analysis of systems in terms of their form, structure &amp; degree of interconnectivity (Küppers, 1999).</td>
</tr>
<tr>
<td>(P8) Feedback</td>
<td>x</td>
<td></td>
<td></td>
<td>Inter-relations between the system parts result in feedback loops where components in the output stage inform components in the input stage (Andriani, 2003).</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>--------------------------</td>
<td>------------------------</td>
<td>-----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>(P9) Pattern recognition/learning</td>
<td></td>
<td></td>
<td>x</td>
<td>As a result of feedback loops adjustment in CAS takes place through the learning experience exhibited by its agents. This will change the agents' effect to realise the required outcome (Webb &amp; Lettice, 2005).</td>
</tr>
<tr>
<td>(P10) Historicity &amp; path-dependence</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>The behavior of the system in one period of time feeds back and informs to determine behavior in the next time; this gives the system ‘historical dimension’ (Stacey et al., 2000).</td>
</tr>
<tr>
<td>(P11) Self-organization &amp; selection</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>Kant (1970) introduced the notion of self-organization as a mechanism to explain the emergence of order in CAS where external influences, e.g., natural forces or social contracts do not govern the internal dynamic of an entity/organization.</td>
</tr>
<tr>
<td>(P12) Coevolution</td>
<td></td>
<td>x</td>
<td>x</td>
<td>Agents rarely are partitioned into non-overlapping groups; they rather participate in multiple neighborhoods/undertakings simultaneously, where their various activities coevolve (Anderson, 1999).</td>
</tr>
<tr>
<td>(P13) Simple rules</td>
<td>x</td>
<td></td>
<td></td>
<td>Holland (1995) defines a CAS as a system composed of interacting agents that respond to external events/stimuli, where agents’ response to triggers/stimulus can be defined in terms of &quot;simple rules&quot;.</td>
</tr>
</tbody>
</table>

Table 1 *CAS Principles*
4. Reflection on CAS to identify whether all its principles have been fully adopted in agile development, or whether there are missing emergence attributes that could be utilized to improve emergence characteristics of agile development.

PRINCIPLES OF AGILE DEVELOPMENT

Agile development refers to a development life cycle designed for a series of methods that have been characterized as “light-weight” methods. These were introduced to replace traditional/heavy-weight development methods in order to cope with change and the need for speed in modern development environments (Highsmith & Cockburn, 2001). The degree of adoption of agile methods is a subject of some debate but in 2005 a Forrester survey suggested that in North America and Europe the figure was around 14% (Schwaber & Fichera, 2005) by 2010 this appeared to have risen to 35% although it is not clear that these figures are directly comparable. Nevertheless, Forrester described this as agile becoming ‘mainstream’ (Krill, 2010). The state of agile development survey VersionOne (2011) with around 6,000 respondents from 80 countries reports a high percentage of agile method adoption in systems development with more than 80% of respondents having adopted agile development practices. Around half of the sample have at least 2 years experience with agile projects and more than a third have up to 5 years experience with agile, indicating some depth of understanding of agile in organizations. In the same survey even large software development projects with over 250 employees showed significant agile adoption rates. Of course it should be recognized that the samples in such surveys by agile product and service providers may well be skewed to those with an interest in agile but nevertheless it is clear that agile is of significant and growing importance in information systems development.

Although the agile manifesto represents an advancement in counteracting the critique of how agile methods work, it is important to note that there is still no universally accepted definition of an agile method (Conboy & Fitzgerald, 2004). For example, Highsmith (2002) defines agility as “the ability of an organization to both create and respond to change in order to profit in a turbulent business environment”. Perhaps the best known interpretation of agile is provided by the manifesto of agile software development (Agile Alliance, 2001), a group statement of 15 representatives of agile methods, which defined twelve principles of agile development as follows:

- Early and continuous delivery of working software;
- Embracing change in requirements;
- Shorter development cycles;
• Collaboration between business people and developers throughout the project;
• Motivated individuals and empowered teams;
• Face-to-face conversation;
• Working software is the primary measure of progress;
• Sustainable development: sponsors, developers, and users should be able to maintain a constant pace indefinitely;
• Good design & technical excellence;
• Simplicity: the art of maximizing the amount of work not done;
• Self-organizing teams: best architectures, requirements, and designs emerge from self-organizing teams, and;
• Reflections & adjustments: at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

It is important to distinguish between flexibility in requirements or in other words the embracing of changes to requirements, as compared to flexibility in software architectures, that covers reusability of architectural building blocks, etc. In light of this discussion and analysis agile principles are distilled and summarized as shown in Table 2.

**INTERPRETING AGILE DEVELOPMENT FROM CAS PERSPECTIVE**

Agile development methods, such as SCRUM, Extreme Programming (XP), SCRUM/XP Hybrid, and other light-weight methods share similar agile practices that are argued to realize agile principles (Table 2). If we are to view agile development from a complexity perspective, it is necessary to examine the principles and practices that these agile approaches adopt and identify if they have the same characteristics and rationales as CAS principles (Table 1). This should help better understand agile principles (see Table 2) and to identify the complexity concepts in action behind these approaches. For example, extreme Programming (XP) adopts practices, such as onsite customer for better communication, small and frequent releases, simple designs, pair programming, time-boxing, continuous integration, code-reuse and re-factoring (Beck, 2000). These practices potentially realize agile principles such as interactions & collaboration, embrace change in requirements, flexibility, quick delivery of working software, etc. that we argue have a resonance from a CAS perspective. Similarly, Adaptive Software Development uses JAD (Joint Application Design workshops), incremental prototyping, time boxing, iterative development and other
People-Oriented/ Empowered & Self-organizing Teams
(motivated individuals, empowered teams, sustainable development, self-organizing teams, etc.)

Communication, Interactions & Collaboration
(collaboration between stakeholders, face-face conversation, interactions, etc.)

Focus on Working Software
Shorter Development Cycles
Embrace Change in Requirements
Frequent Delivery of Products (Releases)
Continuous Reflection & Adjustment/Iterative Development
Flexibility of Software Architecture
Simplicity
Technical Excellence

Table 2 Agile Development Principles

agile practices (Highsmith, 2000). The widely-used agile practices and their inter-re-
lations to CAS principles will be provided in the following analysis. This is done based
on the identified principles of agile practices of Table 2. The outcome of this analysis
is provided in Table 3.

People-Oriented/Empowered, Self-organizing Teams
The effective use of people achieves maneuverability, innovation and adaptation,
which supports the principle of “self-organization”. Employees and customers in agile
organizations need to be knowledgeable, empowered and self-motivated (Highsmith
& Cockburn, 2001). Agile practices that operationalize these concepts cover delegato-
ry management (Fowler, 2005) that give team members space to react independently,
in contrast to traditional methods. Beck (2000) introduces the concept of collective
ownership, 40hour/week and open work area he argues will raise individual commit-
tment to the collective benefit and team working (Martin & Martin, 2006). Task boards,
Kanban boards (brainstorming boards with user stories and requirements) and Burn-
down charts (timeline of development activities) (Cohn, 2004) (VersioneOne, 2011) are
agile tools that will support self-organizing teams in steering their activities.

Communication, Interactions & Collaboration
According to Truex et al. (1999) IS emergence is driven by constant social nego-
tiation and communication. In this way decision processes and outcomes emerge
through interaction with the problem context and with stakeholders. This implies
that agile development provides elements of social construction that facilitate the CAS principle of “interaction”. For example, agile practices support communication and collaboration represented in agile modeling sessions (Ambler, 2002a), user participation, onsite customers, focus groups, prototyping and pair programming (Beck, 2000), and others.

Focus on Working Software
Highsmith (2002) and Baskerville & Pries-Heje (2004) emphasize the use of components and the use of integrated CASE tools as a way to focus on the quick delivery of working software. This can be explained in complexity terms where emergence of order happens through integration and reinforcement of local arrangements and structures to form global structures and the stability of their mutual reproduction (Küppers, 1999). In fact, the break down of the software into working components will speed up the development process so that it is more responsive to change (“unpredictability” principle), as well as facilitating the “coevolution” of the different parts of the system towards maximizing the benefits or profit of the organization, and in that way the whole will be greater than the sum of its parts (“non-linearity” principle).

Shorter Development Cycles
Short-cycle time systems development is achieved through shorter development cycles and release-oriented parallel development (Baskerville & Pries-Heje, 2004). This assures fast movement, small scale orientation and thus quick response. This is important as due to the “unpredictability” principle of CAS, future conditions cannot be predicted and therefore it is important to respond quickly to current situations and address imperatives before they become obsolete.

Embrace Change in Requirements & Frequent Delivery of Products
Baskerville et al. (1992) suggest fast movement based on small releases and short-term planning for short-term needs. This refers to the “unpredictability”, “feedback” and “learning” principles in CAS terminology. For this reason planning games that are based on identifying short-term plans is proposed (Beck, 2000), as well as Sprint Back logs and Sprint Planning meetings that cover planning over 30 consecutive calendar days (Martin & Martin, 2006). These plans will evolve and become adjusted with time to the particular context, which maps to the “adaptation as fit to environment” principle of CAS.
Continuous Reflection & Adjustment/Iterative Development

Feedback and adjustment are important elements of agile development, for example Schwaber (2002) suggests iterative development and Highsmith (2000) suggests learning loops. From a complexity lens this supports the “feedback” and “learning” principle. Through frequent delivery of products (frequent releases) developers know when deviation from the goal has occurred and thus make any necessary corrections, which maps to the “edge of chaos” principle. Highsmith (2000) suggests development to follow “speculate-collaborate-learn” that he refers to as an adaptive cycle, with the aim to adapt to ever-changing environments (“adaptation as fit to environment” CAS principle).

Flexibility of S/W Architecture

Component-based development and code re-factoring are suggested to facilitate flexibility of software code and architecture (MacCormack, 2001; Highsmith, 2002; Beck, 2000). This in turn decreases the degree of “connectivity and interdependence” within systems components. In CAS agents’ interactions result in networks that will then morph into high-level arrangements that will enable complex structures and processes to coevolve (Lichtenstein & McKelvey, 2004). This facilitates adaptive behavior as supported by CAS principles.

Simplicity

The Agile Alliance (2001) argues for simplicity that they define as the art of maximizing the amount of work not done, which is essential for rapid response and adaptation. Highsmith (2002) identifies three facets of simplicity, simplicity as minimalism e.g., minimal documentation, remove redundancy, e.g., cleaned-up sketches etc., simplicity in design, e.g., simple designs, feature-based designs, abstractions and simplicity in strategy e.g., mission-focused plans, etc. This supports the CAS principle of “simple rules”. This can also be explained as simplicity enhances diversity that creates more possibilities to react flexibly as agent networks combine the most different variants, characters and functions (Webb et al., 2004).

Technical Excellence

Highsmith (2002) argues that good discipline is still required in order to be flexible without descent into anarchy, which represents the CAS concept of “edge of chaos”. Cockburn (2000) suggests quality controls; these can be realized by quality reviews and testing. Baskerville et al. (2002) identify practices to balance quality and agility;
including frequent peer and customer reviews and frequent unit, integration & regression testing. Boehm (2002) argues that agile methods need to be applied with care, this implies that minimal use of development methods or generative development rules are still required (Cockburn, 2000, Highsmith & Cockburn, 2001).

A mapping of the literature that covers agile development and their inter-relation to CAS principles as identified in this section is provided in Table 3.

CONCEPTUALIZING AGILE DEVELOPMENT AS A COMPLEX ADAPTIVE SYSTEM

Practices of agile development have been found to be the operationalization of the complexity concepts as discussed in the previous section. For example, according to CAS principles, interactions and social construction are critical intangible drivers for change and adaptation, as supported by agile practices like JAD (Joint Application Design) workshops, agile modeling sessions, prototyping, pair programming and others. This is beside development practices that support short-term orientation, speed in development and flexibility, such as short-term plans, time-boxing, shorter development cycles, focus on working software, etc. that operationalize CAS principles of unpredictability, non-linearity, connectivity & interdependence and co-evolution. The concept of edge of chaos is also particularly highlighted as it ensures reliability and rigor of agile development such that it will not lead to chaotic situations. Edge of chaos represents the balance between strict adherence to structure and absolute freedom, and from the above analysis it relates to continuous reflection, adjustments, and technical excellence principles. These are represented in iterations, learning loops, frequent-releases, as well as testing techniques, customer reviews and minimal generative rule-type methods.

Thus, via this analysis it is argued that the identification of the CAS principles that have been adopted in agile IS development (Table 3) provides theoretical support to agile development. Next, this is built upon to achieve a generic representation of complex adaptive information systems referred to as CAIS (Complex Adaptive Information Systems). This provides a theoretical representation of CAIS (Figure 2) showing the means and detail by which modern IS development methods like agile implicitly utilize the complexity concepts to enable emergence and cope with change.

CAS theory implies that agents that form the system continuously interact with each other and with the global environment to form stable, global patterns that suit the current settings in the system and the environment as depicted in Figure 1. The
emergence process is governed by CAS principles; unpredictability, non-linearity, diversity, interactions, edge of chaos, adaptation as fit to environment, system interconnectivity, feedback, learning and pattern recognition, historicity and path dependence, self-organization, coevolution and simple rules. Similarly, agile information systems development can be conceptualized using CAS where agents that represent project stakeholders, development team, users and software system components continuously undergo interactions to adapt to changing requirements. The mapping of CAS principles to agile development practices shows that all reviewed agile practices surveyed from literature support CAS, but it was found that some CAS principles are less well developed in agile and are not fully addressed in agile practice. Figure 2 provides an interpretation of agile development as a CAS as it shows CAS principles and agile practices that support them (attached with the principle). The CAS principles that are not fully addressed in agile are highlighted in bold and they have vague agile practices attached to them, for example empowered teams, flexibility of software architecture, embrace change in requirements, etc., or no attached practices. In the following we will suggest how to introduce new agile practices to support not well spelled out CAS principles. Furthermore, as shown in Figure 2, CAS principles can be classified as related to environment, like unpredictability and adaptation as fit to environment, as related to agent properties, like diversity, non-linearity, connectivity and simple rules, as related to selection like self-organization, coevolution and edge of chaos, and as related to iterations and feedback, learning and pattern recognition, etc. In future studies we will investigate how such grouping could provide a further high-level abstraction of CAIS theory and underlying agile practices.

The not spelled out CAS principles are; diversity, inter-connectivity, self-organization, adaptation as fit to the environment, pattern recognition, and historicity and path dependence, and are now discussed in further detail:

- The principle of diversity needs to be further addressed in agile development, as currently it is just represented in the simplicity approach. Diverse interest and knowledge will stimulate agents to explore different regions in the system’s state space (Heylighen, 2001). This can happen through variant skill development within agile teams and supportive training workshops.

- According to Axelrod and Cohen (1999) adaptability is the outcome of a selection process that leads to an improvement according to some measure of success where current events heavily influence the probabilities of later events (historicity & path dependence). Pattern recognition implies that learning is an essential part of adaptive development. Thus, it is important to utilize previous experiences and
<table>
<thead>
<tr>
<th>Agile Principles</th>
<th>Agile Practices</th>
<th>Supportive IT Literature</th>
<th>Supportive CAS Principles</th>
<th>Interpretation from CAS Perspective</th>
</tr>
</thead>
<tbody>
<tr>
<td>self-organizing,</td>
<td>40 hour/week Open work area Task board, Kanban Board, Burndown charts</td>
<td>Beck (2000)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Communication,</td>
<td></td>
<td>Cohn (2004)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interactions &amp; Collaboration</td>
<td>User participation, Onsite customers</td>
<td>Truex et al. (1999) Highsmith (2002)</td>
<td>Interactions (P4)</td>
<td>Agile principles that emphasize communication &amp; collaboration are grounded in the CAS principle of “Interactions between agents”. This provides the evidence why practices like user participation, participative design, pair programming, etc., facilitate a mean of interactions and coordination, and thus leverage emergent response.</td>
</tr>
<tr>
<td>Focus on working</td>
<td>Daily stand up meetings Focus groups JAD (Joint Application Design) Agile</td>
<td>Martin (1991)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Focus on working</td>
<td>Customization, Reusability Components Integrated CASE tools</td>
<td>Highsmith (2002) Baskerville &amp; Pries-Heje (2004)</td>
<td>Unpredictability (P1)</td>
<td>Break down of the software into working components will speed up the development process so that it becomes more responsive to change (“unpredictability” principle), as well as facilitate the “coevolution” of the different parts of the system and in that way the whole will be greater than the sum of its parts (“non-linearity” principle).</td>
</tr>
<tr>
<td>S/W</td>
<td></td>
<td></td>
<td>Nonlinearity (P2)</td>
<td></td>
</tr>
<tr>
<td>Coevolution</td>
<td></td>
<td></td>
<td>Coevolution (P12)</td>
<td></td>
</tr>
<tr>
<td>Agile Principles</td>
<td>Agile Practices</td>
<td>Supportive IT Literature</td>
<td>Supportive CAS Principles</td>
<td>Interpretation from CAS Perspective</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------</td>
<td>--------------------------</td>
<td>---------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Embrace Change in Requirements</td>
<td>Short-term planning, Planning game, Sprint Back logs</td>
<td>Baskerville et al. (1992), Beck (2000), Martin &amp; Martin (2006)</td>
<td>Unpredictability (P1), Adaptation as fit to environment (P6)</td>
<td>Embrace change in requirements as reflected in short-term planning will improve adaptation and adjustment to fast changing environments. This is in contrast to following a pre-plan with well defined long-term assumptions that could obsolete soon.</td>
</tr>
<tr>
<td>Frequent Delivery of Products</td>
<td>Small releases, Increments, Continuous integration</td>
<td>Highsmith (2000), Martin &amp; Martin (2006)</td>
<td>Unpredictability (P1), Feedback (P8), Learning (P9), Adaptation as fit to environment (P6)</td>
<td>Through frequent delivery of software represented in small releases &amp; increments the development team can assure quick and gradual response. This will facilitate feedback, learning, and adjustment to unpredictable requirements.</td>
</tr>
<tr>
<td>Continuous reflection &amp; adjustment / Iterative development</td>
<td>Iterations, Learning loop, Adaptive cycle, Iteration planning, Daily Stand up meetings, Frequent-releases, Learning environment (pilot projects, prototyping, focus groups), Continuous re-development</td>
<td>Schwaber (2002), Highsmith (2000), Beck (2000), VersionOne Survey (2008) (2011)</td>
<td>Feedback (P8), Learning (P9), Edge of chaos (P5)</td>
<td>Through iterative development &amp; frequent software releases developers adjust the product at the end of each iteration/project cycle. Based on this next, subsequent iterations or releases will be guided. This supports “feedback” and “learning” principles. It also reflects “edge of chaos” concept; i.e. deviations from the goal (rationale of the system) can be monitored and necessary corrections can be taken.</td>
</tr>
</tbody>
</table>

Table 3 Agile Principles, Related Agile Practices & Supportive CAS Principles
<table>
<thead>
<tr>
<th>Agile Principles</th>
<th>Agile Practices</th>
<th>Supportive IT Literature</th>
<th>Supportive CAS Principles</th>
<th>Interpretation from CAS Perspective</th>
</tr>
</thead>
</table>

**Table 3** Agile Principles, Related Agile Practices & Supportive CAS Principles (continued).
accumulative knowledge as part of agile development and not let it happen by
chance. In this regard, Dabrowski et al. (2011) suggest the codification of informal
notes, brainstorming ideas, instant messaging discussion topics and other tacit
knowledge items as a way to trace events’ history and support agile decision
making activities.

• The principle of adaptation as fit to environment needs to be further harnessed
in agile development as currently it is just represented spontaneously in planning
games and sprint planning. Contextual analysis techniques need to be included
in agile requirements specification beside business process modeling techniques
that focus on data and process modeling. A global requirements exploration tool
was introduced by Alaa, et al. (2006) to help outline high-level requirements that
a typical e-business project could encounter. This guided analysts in adapting to
various project contexts.

• It is noted in the literature that agile development often faces personal barriers,
represented in shortage of mid-level management capable of playing the role
of agile champions. Traditional high-level management typically feels a loss of
control in delegating decisions, and traditional developers will resist taking extra
responsibilities associated with hybrid teams and collective ownership (VersionOne,
2011; Karekar et al., 2011). In this regard it is suggested that training workshops are
used to leverage collaboration and delegatory management skills by improving
collaboration, pattern recognition and thus self-organization capabilities.

• Emphasis on inter-connectivity of system components and flexibility of
architecture has attracted attention in some agile literature. For example, Faber
(2010) introduces the concept of skeletal design where system architects quickly
decide on system sub-components with their interfaces and messaging to ensure
architecture loose coupling. These techniques have not yet been formally adopted
into agile methods, as they might conflict with agile principles that focus on coding
and the quick delivery of working software that might hinder the extensive design
of software architecture (Abrahamsson et al., 2010).

CONCLUSIONS

The research presented in this paper sits in the mainstream of theoretical and
explanatory Information Systems (IS) change models. IS emergence phenomena was conceptualized by building on a profound theoretical framework, that
of complex adaptive systems theory (CAS). This analysis identified IS emergence
characteristics that would make information systems development more responsive
to change. It is argued that the derived framework (Table 3) represents a theoreti-
cal foundation for agile IS development, based on CAS principles, that are found to facilitate fast response and emergence. This is a significant improvement on most development methods that in practice simply hope that certain beneficial emergent properties will happen, without providing a theoretical basis or evidence. Based on this analysis a Complex Adaptive Information System (CAIS) model is suggested that can help analyze emergence attributes in complex software projects, as it identifies generic IS emergence mechanisms supported by CAS (Figure 2). This provides a basis for reliable guidelines to help various software projects realize emergence and agility.

The framework also highlights areas that might usefully be improved by the agile community based on better utilization of CAS principles (attributes in bold in Figure 2). These emergence characteristics are currently missing in the practice or not spelled out clearly. In particular the research identifies possible agile development improvements based on:

- Diversity building techniques and workshops
- Skill development for delegatory management and self-organization
- Global contextual analysis techniques
- Coarse-grained software architecture design techniques
- Codification of unstructured tacit knowledge, represented in notes, instant messaging feedback and opinions, etc. that would support pattern recognition, historicity and path dependence.

These additions we argue are of particular importance as agile development is increasingly widely adopted, even in large-scale global software development projects (VersionOne, 2011). This research is ongoing and future work will focus on refining and validating the generic CAIS framework.

REFERENCES


Figure 2 Toward Complex Adaptive Information System (CAIS) (Reflects IS Emergence Characteristics of Agile Development)


**Ghada Alaa** is currently full-time Software Architecture Specialist at the Egyptian Cabinet’s Information and Decision Support centre. Prior to that she held posts as Assistant Professor at the School of Informatics and Computer Science, the British University in Egypt (2008-2011) and R&D Manager at the Information Technology Institute, the Ministry of Communications & Information Technology, Egypt (2007-2008). She holds a Ph.D. in Web-based Systems Development & Evolution, Brunel University, UK (2006). Her research interests lie in the area of Web development methodologies in particular agile, and software engineering design frameworks, evolution processes and metrics for modern Web applications to include Web 2.0, Service-oriented architecture (SOA) and cloud computing.
Guy Fitzgerald is Professor of Information Systems in the School of Business and Economics at Loughborough University, UK. He has also worked at Brunel University, Birkbeck College, University of London, Oxford University and Warwick. His research interests are concerned with the effective management and development of information systems and he has published widely in these areas. He has undertaken a number of cases studies in organizations that have used information systems to enable significant organizational transformation. He has also undertaken research in relation to strategy, executive information systems, outsourcing, and flexibility. He is founder and coeditor of the Information Systems Journal (ISJ) from Blackwell/Wiley. He is also author (with David Avison) of a major text from McGraw-Hill entitled Information Systems Development: Methodologies, Techniques and Tools, now in its Fourth Edition. He has been President of UKAIS and Vice-President (Research) of AIS (the Association for Information Systems).