

On Discovery and Display of Agile Security Patterns

Rick Dove

Stevens Institute of Technology
Hoboken, NJ, USA
rick.dove@stevens.edu

Laura Shirey

Department of Defense
Fort Meade, MD, USA
laurashirey@msn.com

Abstract

This article explores the nature of agile security concepts and techniques, and casts them as system architectural patterns. The work reported here provides an experimental descriptive format for patterns of agile security, and develops three examples that exercise the pattern format and demonstrate agile security concepts. The examples are inspired by multiple instances of specific security approaches described in the literature, and have been abstracted for general concepts employable in a broader context. The work was undertaken as an experimental foundation for characterizing agile security by identifying and cataloging exemplar patterns on a broader scale, intended as the initial platform for a project of the International Council on Systems Engineering (INCOSE) working group on System Security Engineering.

Introduction

Agile security refers to security approaches that effectively mirror, as a minimum, the agile characteristics exhibited by the system attack community: self organization, adaptable tactics, reactive resilience, evolvable strategies, proactive innovation, and harmonious involvement. In this respect, agile security is not the current norm in deployed system security. It is the intent of this article to provide some examples of agile security to facilitate the development of common concepts and models.

The growing body of work in patterns for system architectures has inspired a pattern descriptive approach arising from reviews of Christopher Alexander's seminal construction-architecture pattern work (Alexander 1977), software design patterns as suggested by (Gamma et al. 1994), security pattern approaches by (Kienzle et al. 2002, Mouratidis et al. 2003, Schumacher et al. 2001 and 2006, Steel et al. 2005, Yoder et al. 1997, and Yoshioka et al. 2008), Robert Cloutier's doctoral work on architectural patterns for complex systems (Cloutier 2006), and others.

A relevant informative history of the work in patterns, with a focus on security patterns, is available in (Hafiz et al. 2006), where concerns and issues of pattern classification and organization are addressed – something that will become important to the cataloging of agile security patterns once a coherent body of patterns takes shape. (Hafiz 2007) carries this theme further, addressing methods for developing a classification scheme.

A literature search on “agile security” reveals a general focus on the agility of security-system development processes, rather than the agility of system security in operation, though (Lipson 2006) states the case well, while (Forrest et al. 1994) and (Parunak 1997) clearly spawned fruitful lines of agile system-security research and development. The focus in this article is on agile-system operation rather than agile system-development. Very little is available on the general nature and characterization of agile security-system operation, so the work presented here should offer a starting platform.

The work presented here was done to provide a starting point for a more extensive and collaborative search for appropriate security examples, and for subsequent convergence on a descriptive format.

In the end, patterns should bridge the communication differences between planners and implementers of systems, offering a common language understandable to both.

Agile Security

Current-generation security is characterized principally as reactive, inventing and deploying in response to the escalating sophistication and success of attack experiences. As after-the-fact defense insertion, security response typically deploys add-on functional subsystems or new external guarding systems, force-fit to the system that needs protection.

In contrast, the innovation of determined adversaries adapts to coexist with reactive defensive measures and evolves to create new methods and select new targets. Looked upon collectively as a self-organizing system of exploitation systems, adversarial activity is driven by intelligent goal-seeking and by unbounded experimentation – preying upon outclassed reactive strategies and new defenseless targets. Adversaries are diverse in nature and allegiance, but their strength is rooted in the six common characteristics listed earlier. In a word, the adversary is agile.

To provide parity with the agility of intelligent attacking systems, mirroring their six agile characteristics, as outlined in a working paper (Dove 2009a), seems appropriate as a minimum:

- [S] Self-organizing – with humans embedded in the loop, or with systemic mechanisms.
- [A] Adapting to unpredictable situations – with reconfigurable, readily employed resources.

[R] Reactively resilient – able to continue, perhaps with reduced functionality, while recovering.

[E] Evolving in concert with a changing environment – driven by vigilant awareness and fitness evaluation.

[P] Proactively innovative – acting preemptively, perhaps unpredictably, to gain advantage.

[H] Harmonious with system purpose – aiding rather than degrading system and user productivity.

In the work reported here we have used these six SAREPH characteristics as filters for selecting candidate agile security techniques – though no research conclusions guide us yet for suggesting how many or what combinations might be minimally necessary, or even if these six are sufficient.

In Search of Agile Security Examples

Stephanie Forrest (Forrest et al. 1994) is a pioneer in adapting lessons from biology to security strategies. Though her focus tends to be on cybersecurity, her insights seem appropriate to a much wider class of systems.

“Among the principles of living systems we see as most important to the development of robust software systems are: Modularity, autonomy, redundancy, adaptability, distribution, diversity, and use of disposable components.” (Forrest et al. 2005)

Life adapts and evolves, it is resilient, it is innovative, and it is in harmony with its environment. Lessons from the sustained evolution of species would seem appropriate for consideration in agile security design.

There are many examples of agile security in the natural world. Several examples are examined in Sagarin and Taylor’s book *Natural Security* (Sagarin et al. 2008) describing lessons taken from nature that can

be applied to security systems. A main premise of theirs is that natural security involves the notion of learning to live with risk rather than declaring war on it. Sagarin and Taylor categorize the approach into three areas: adaptation, resource allocation, and understanding risk.

Sophie Wilkinson, in an editorial piece for Chemical and Engineering News (Wilkinson 2001), and Bruce Schneier, in a security trade magazine interview (Berinato 2008), discuss some interesting examples of security that occur in plants, trees, and animals. Many plants, for instance, have the ability to emit volatile chemicals in attempts to dissuade insects from feasting on their leaves. Some, like the lima bean, even send out signals to insect predators to come to the rescue and eliminate the threat, and signal other plants in the local area to do the same.

Well known are the threat-responsive swarming behaviors exhibited by ants and bees, where large numbers of simple units work together to drive off or eliminate the threat. As individual units they only understand a simple common immediate goal and follow basic rules to achievement.

Social animal life exhibits built-in systemic mechanisms for detecting security-threatening behavior among its members, and mitigating that behavior if it is evaluated as intolerable. Peer behavior policing is evident in humans (Myers 2008), animals (Flack et al. 2006) and insect societies (Heinze 2003).

These threat-responsive techniques in the natural world offer insight into potential agile techniques that can be applied to the threats aimed at our systems. The example patterns shown in this article were clearly inspired by these biological techniques.

Pattern Description Format

A variety of in-use pattern formats were considered for this project. The key pattern formats explored were mentioned earlier; with additional on-line repositories investigated

and shown in the reference section (undated) as (Hillside Group), (Portland Pattern Repository), (SOA Patterns), and (Microsoft).

While all of these different but similar formats suit their individual purposes, none fit the agile security needs sufficiently; where dynamics and forces in balance play key roles in driving the expression of SAREPH characteristics, and referenced examples are particularly necessary in this preliminary stage of pattern exploration. Nevertheless, Figure 1 shows the adapted format rooted in Christopher Alexander’s context-problem-solution framework.

Table 1: Pattern Description Format

Name: Descriptive name for the pattern.
Context: Situation that the pattern applies to.
Problem: Description of the problem.
Forces: Tradeoffs, value contradictions, key dynamics of tension and balance, constraints.
Solution: Description of the solution.
Graphic: A depiction of response dynamics.
Examples: Referenced cases where the pattern is employed.
Agility: Evidence of SAREPH characteristics that qualify the pattern as agile.
References: Literature access for examples.

Three Examples

Recognizing patterns in security strategies, identifying those that are agile, and defining them in a reusable pattern format can help accelerate the inclusion of agile security as part of the systems engineering process. This initial work should provide a platform for subsequent study and augmentation.

Three examples of agile security patterns were selected for an initial trial of this display format and the SAREPH qualifications. In each case pattern profiles are abstractions of specific referenced cases in the literature.

Table 2: Pattern – Dynamic Phalanx Defense

Name: Dynamic Phalanx Defense
Context: a stationary or mobile asset subject to unpredictable swarm attacks.
Problem: Attackers can come in many and unpredictable forms, and in virtually unbounded quantities, with no advance warning. For instance, A DDoS attack on an Internet service node may be of many different types; an attack on a naval asset may be surface, undersea or air in many different varieties.
Forces: Resilience of service vs. cost of service. Comprehensive counter capability and capacity vs cost and disharmony of a broad standing counter force.
Solution: the ability to detect the threat and the nature of its attack, the ability to produce and deploy appropriate disposable counter-measures, the ability to deploy measure-for-measure and to stand down or dispose of deployed counter measures when the threat is vanquished.
<p style="text-align: center;">Aggressive shield waxes and wanes measure-for-measure in real time</p>
<p>Example: Artificial immune system – detection, selection, cloning and retirement applied to mobile network intrusion detection and repulsion. See (Edge et al. 2006, Zhang et al. 2008).</p> <p>Example: Botnet denial of service defense – Instantly recruit an unbounded network of computers to shield a server from being overwhelmed by botnets. See (Dixon et al. 2008, Mahimkar et al. 2007).</p> <p>Example: Just-in-time drone swarms – Load disposable drones with modular sensor and weapon choices, and deploy quantities as needed. See SWARM, JITSA discussion in (Hambling 2006).</p> <p>Example: Plant chemical defense – Insect saliva triggers selective toxic gene expression and gas emissions that call in selective insect predators. See (Wilkinson 2001).</p>
<p>Agility: Self organization grows and shrinks a counter swarm in measured response to an attack swarm. Adaptability selects appropriate counter-swarm agents from modular resources. Resilience is exhibited with expendable and replicable counter agents, and in continued operation of the protected asset, though perhaps at reduced performance. The process is harmonious with protected asset functionality as it is only activated upon detecting a threat, and then only in dynamic measure-for-measure as needed. [S-A-R-H]</p>
<p>References: (see reference section, only URL shown here, all accessed 30Nov09)</p> <ul style="list-style-type: none"> • (Dixon et al. 2008) www.cs.washington.edu/homes/ckd/phalanx.pdf. • (Edge et al. 2006) http://paper.ijcns.org/07_book/200603/200603C08.pdf • (Hambling 2006) http://defensetech.org/2006/04/10/drone-swarm-for-maximum-harm/ • (Mahimkar et al. 2007) www.cs.utexas.edu/~yzhang/papers/dfence-nsdi07.pdf • (Wilkinson 2001) http://pubs.acs.org/cen/critter/plantsbugs.html • (Zhang et al. 2008) www.computer.org/portal/web/csdl/doi/10.1109/ICNC.2008.782

Table 3: Pattern – Peer Behavior Monitoring

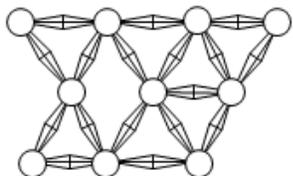
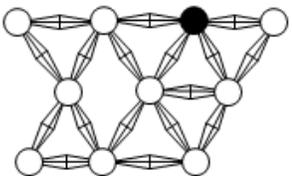
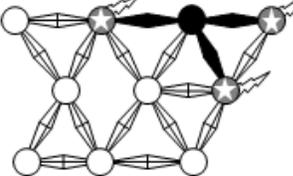
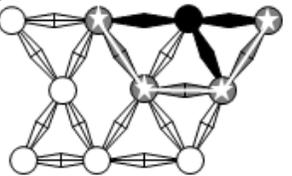
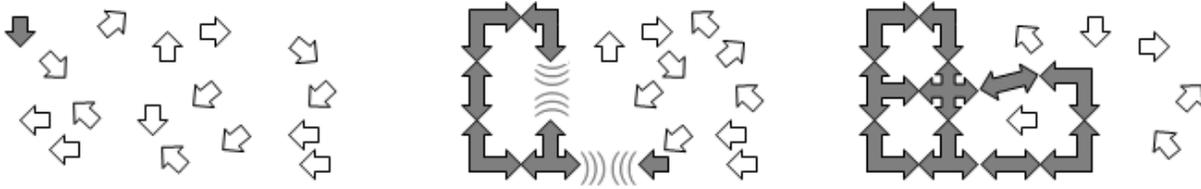
<p>Name: Peer Behavior Monitoring</p>
<p>Context: a collection of independent systems (or agents) selectively interoperating as a system of systems (or multi-agent system) in group-cooperative pursuit of common critical mission.</p>
<p>Problem: Any member of a group may threaten group mission accomplishment through malicious intent, malfunction, or misconfiguration. Detecting and mitigating threats and potential threats when they arise requires comprehensive activity monitoring and evaluation. The quality of centralized monitoring, evaluation, and mitigation activity may be compromised unacceptably by high frequency and high quantity of relevant sensor data.</p>
<p>Forces: Political considerations of centralized vs decentralized control. Breadth and depth of local situational knowledge vs remote partial knowledge and assumptions. Broad strategic vs local tactical knowledge. Sense and response time constraints vs communication bandwidth.</p>
<p>Solution: Group members selectively monitor peers within a span of local relevance, comparing aspects of actual behavior against internal models of acceptable behavior. Detection of deviance triggers actions for evaluation and mitigation. Peer monitoring takes advantage of local and situational knowledge available to each member, with comprehensive behavior detection and evaluation distributed among many members. Behavior outside acceptable limits may be reported to a central control for evaluation and mitigation, or may be resolved locally.</p>
<div style="display: flex; justify-content: space-around; align-items: flex-end;"> <div style="text-align: center;">  <p>peer behavior observation</p> </div> <div style="text-align: center;">  <p>a peer goes bad</p> </div> <div style="text-align: center;">  <p>some peers note & report</p> </div> <div style="text-align: center;">  <p>...or...decide locally</p> </div> </div> <p>Peers monitor for aberrant behavior and tattle, or decide locally on need and mode for action</p>
<p>Example: Autonomous swarming weapon systems – Ubiquitous peer behavior evaluation among unmanned autonomous weapon systems on swarm strike missions. See (Dove 2009b).</p> <p>Example: Ad hoc networks – Self policing of misbehavior in mobile ad-hoc networks with local consensus evaluation and isolation of offending nodes. See (Buchegger 2005).</p> <p>Example: Self-organizing multi-agent systems – Unilateral evaluation and execution of an offending agent by a monitoring agent, requiring monitoring-agent suicide to avoid repetitive malicious and faulty monitor decisions. See (Clulow 2006).</p> <p>Example – Electric power grid – Distributed agents detect attacks and malfunctions of specific nodes, invoking contingencies based current agent relationships. See (Smathers et al 2001).</p>
<p>Agility: Agents self reorganize and adapt according to observed and evaluated behaviors of other agents. Rapid offender removal and relationship replacement provides high resilience. Proactive decision making on suspicious and precursor behavior avoids attacks before they occur. Monitoring does not interfere with individual task accomplishment, avoids central decision delays, and has been shown to provides superior decision accuracy. [S-A-R-P-H]</p>
<p>References: (see reference section, only URL shown here, all accessed 30Nov09)</p> <ul style="list-style-type: none"> • (Buchegger 2005) http://infoscience.epfl.ch/record/52175/files/BucheggerL05.pdf • (Clulow 2006) http://people.seas.harvard.edu/~tmoore/OSR-suicide.pdf • (Dove 2009b) www.parshift.com/Files/PsiDocs/Pap090901IteaJ-PathsForPeerBehaviorMonitoringAmongUAS.pdf • (Smathers et al 2001) http://certs.lbl.gov/pdf/SAND00-1005.pdf

Table 4: Pattern – Swarming Threat Sensors

Name: Swarming Threat Sensors
Context: a critical space, fixed or mobile, too large to blanket completely with threat-sensor coverage.
Problem: When blanket fine-grained sensor coverage is impractical, comprehensive monitoring of an asset-containing area can be impossible if the protected area is large, if the area changes shape or is mobile, or if mobile threats can probe and find blind sensor spots.
Forces: Quantity of sensors vs. sensor cost. Identical multi-purpose sensors vs. distributed diversity of specialty sensors.
Solution: Use autonomous mobile sensors exploring at random until a threat is detected, then self organizing in focused swarm behaviors to map and track dynamic threat activity.
 <p style="text-align: center;">Swarm convergence seeks optimal sensor distribution to monitor detected threat</p>
<p>Example: Hazardous leak – Monitor an area for hazardous leaks. When any agent detects a leak, swarming occurs to map and monitor the leak’s extent and spread. Autonomous repositioning control for each robotic sensor continually adjusts to its neighbor’s location and detection status to improve overall coverage through iterative convergence. See (Parunak et al. 2004).</p> <p>Example: Naval defensive perimeter – Small inexpensive UAV (unmanned aerial vehicle) escorts monitoring vast mobile threat area of naval ships on maneuver. See (Zoccola 2001) for an early concept discussion of since advanced naval SWARM project.</p> <p>Example: Wireless sensor network: Individual mobile sensor nodes survey their neighbors and continually adjust their location, converging on optimal total coverage. See (Lee 2007).</p>
<p>Agility: Mobile sensors continually self organize their location distribution to find optimal coverage across the total monitored area. Sensor coverage can adapt to threat events within the covered area and to changes in shape and location of a mobile area. Resilience is exhibited when coverage once provided by a failed sensor is taken up by location adjustments of neighbor sensors. Proactive random hunting can put previously safe intruders into reactive defensive mode. Mobile sensing as referenced here does not interfere with the system under protection and can harmoniously improve system functionality by better and earlier discovery of potential system impairments. [S-A-R-P-H]</p>
<p>References: (see reference section, only URL shown here, all accessed 30Nov09)</p> <ul style="list-style-type: none"> • (Lee 2007) http://paper.ijcsns.org/07_book/200712/20071210.pdf. • (Parunak et al. 2004) www.newvectors.net/staff/parunakv/ANS04.pdf • (Zoccola 2001) www.dt.navy.mil/pao/excerpts%20pages/2001/UAV3_01.html

Discussion

Finding a suitable pattern format, or *form* as it is called in the patterns community, was a major objective. Many examples of form from different domains were explored, but so was literature that discusses the issues of constructing pattern forms (Salustri 2005) and pattern languages (Meszaros 1996 and Salingaros 2000). These excellent discussions cannot be synopsized here, but their guidance affected this project in both its convergence on the employed format and on the issues of forward planning for how this format and these examples might best serve subsequent phases of work as we move toward a pattern language for agile security.

The three examples shown in this article exhibit only five of the six possible SAREPH characteristics – evolution (E) not being present in any of them. The nature of the self organization (S) exhibited by all three is systemic, none of them relying upon embedded humans to manage the self-organizing activity. Examples of system security that evolves systemically, without human management, were not abundant in the literature search conducted for this project, (Edge 2006) being a notable exception.

Though there is reasonable comfort that the three patterns displayed in this article qualify as agile security patterns, and that the pattern descriptive framework evolved sufficiently during the project to be effective, both qualification and format are expected to benefit from additional experimental use and evolution.

This article focuses on pattern form rather than on imparting comprehensive pattern understanding. Full appreciation of the three example patterns relies upon knowledge and interpretation of the references cited as support for each example pattern.

Reflecting on the pattern forms as stand alone descriptions, it is felt that the “agility” profile, for instance, might describe the agile

qualifications of the pattern in more detail – to clearly explain the essence of methods that distinguish a particular pattern from any other. But it is also felt that the content of the pattern form should be a synopsis, and not an exhaustive education. In these early times, when we are seeking path finder patterns, a single pattern candidate could be better explained in a dedicated and separate article treatment, with the pattern form serving as a wrap-up condensation of key points explained in the accompanying text.

Of particular concern throughout the project was the nature of the pattern graphic. It is felt that a graphic has a powerful capability to reveal the essence of a pattern in a memorable way. This is supported by the dominance of visual cortex real estate in the brain, as well as the popular metaphorical belief that a picture is worth a thousand words. However, agile security patterns are based on dynamic activity, not static snapshots. We tried flow charts as graphics to no satisfaction. Time sequenced cartoon panels, or even single panel cartoons that depict action, seem more effective at conveying the essence of dynamics.

Concluding Remarks

In the course of conducting this work, other next-step research became evident. While this initial effort explored patterns of autonomous agents and swarming examples, there are many other concepts that are appropriate and available for similar treatment. For instance, a reasonable body of work already exists in security approaches modeled on artificial immune systems (Forrest et al. 2001, Garrett et al 2005).

The SAREPH characteristics will benefit from more depth of development, and some validation against the characteristics exhibited by the adversary communities.

Pattern languages have pattern hierarchies. Christopher Alexander’s work, for instance, recognizes towns, buildings, and construction

as successively nested pattern categories. In the course of selecting and developing the three examples in this article it became evident that they are all in a middle category of at least a three level hierarchy. A lower level might encompass patterns within individual agents that enable and compel their participation in multi-agent patterns like those in this article's examples. A higher level might encompass patterns of governance, autocatalysis, behavior attractors, and evolution drivers. Are there only three natural levels or are there more to be explored, and how are they bounded? What are representative examples of lower and higher level patterns?

A fair body of work already deals with artificial immune systems. Even a cursory introduction shows clearly that there are many interesting sub-patterns within the general "immune system" pattern. Does this provide a clear boundary within which a relatively small and precise number of sub-patterns might be identified and described; what would they be; and how do they look?

This work was undertaken to provide a starting platform for a subsequent project of the International Council on Systems Engineering (INCOSE) System Security Engineering working group. That project intends to develop a population of "path finder" patterns of agile security, expecting they will be replaced with a more historical-based pattern compendium once sufficient experience is accumulated on the way toward a reasonable "pattern language".

References

- Alexander, Christopher, *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, 1977.
- Berinato, Scott and Schneier, Bruce, Q&A: The Endless Broadening of Security, CSO magazine, June 2008.
- Buchegger, S. and Boudec, J.-Y.L., Self-Policing Mobile Ad-Hoc Networks by Reputation Systems, IEEE Communications Magazine, pp. 101–107, July 2005.
- Crosbie, Mark and Spafford, Gene, Defending a Computer System using Autonomous Agents, Technical Report No. 95-022, COAST Laboratory, Dept. of Computer Sciences, Purdue University, 1994.
- Cloutier, Robert, Applicability Of Patterns To Architecting Complex Systems, Doctoral Dissertation, Stevens Institute of Technology, Hoboken, New Jersey, 2006.
- Clulow, Jolyon and Tyler Moore, Suicide for the Common Good: a New Strategy for Credential Revocation in Self-Organizing Systems, SIGOPS Operating Systems Review, 40(3):18–21, 2006.
- Dixon, Colin, Anderson, Thomas and Krishnamurthy, Arvind, Phalanx: Withstanding Multimillion-Node Botnets, NSDI'08: Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, April 2008.
- Dove, Rick, Embedding Agile Security in Systems Architecture, Insight 12 (2): 14-17, International Council on Systems Engineering, July, 2009.
- Dove, Rick, Paths for Peer Behavior Monitoring Among Unmanned Autonomous Systems, *The International Test and Evaluation Association Journal*, 30 (3): 401–408, 2009.
- Edge, Kenneth S., Gary B. Lamont, and Richard A. Raines, Multi-Objective Mobile Network Anomaly Intrusion, *International Journal of Computer Science and Network Security*, 6(3b):187-192, March, 2006.
- Flack, J. C., Girvan, M., de Waal, F. B. M. and Krakauer, D. C., Policing Stabilizes Construction of Social Niches in Primates, *Nature*, 439 (7075): 426-429, 2006.
- Forrest, S., Perelson, A. S., Allen, L., and Cherukuri, R., Self-Nonself Discrimination in a Computer, In Proceedings IEEE Symposium on

- Research in Security and Privacy, Oakland, CA., May 16–18, 1994.
- Forrest, S. and Hofmeyr, S., Engineering an Immune System, *Graft* 4(5):5-9, 2001.
- Forrest, S., Balthrop, J., Glickman, M. and Ackley, D.. K. Park and W. Willins Eds. *The Internet as a Large-Scale Complex System*, Oxford University Press, 2005.
- Gamma, E., Helm, R., Johnson, R. and Vlissides, J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, 1994.
- Garrett, Simon, How Do We Evaluate Artificial Immune Systems?, *Evolutionary Computation* 13(2): 145-178, 2005.
- Hafiz, Munawar and Johnson, Ralph E., Security Patterns and Their Classification Schemes, Technical Report for Microsoft's Patterns and Practices Group, Sept., 2006.
- Hafiz, M., Adamczyk, P., and Johnson, R. E., Towards an Organization of Security Patterns, IEEE Software Special Issue on Software Patterns, 24(4):52-60, 2007.
- Hambling, Dave, Drone Swarm for Maximum Harm, Defense Tech. April 10, 2006. <http://defensetech.org/2006/04/10/drone-swarm-for-maximum-harm/>
- Heinze, Jürgen, Reproductive Conflict in Insect Societies, In *Advances in the Study of Behavior*, ed. P. Slater, and J. Rosenblatt, 34: 1-57. New York: Academic Press, 2003.
- Hillside Group, PLoP (Pattern Languages of Programs) sponsor, www.Hillside.net.
- Karlin, J., Forrest, S., and Rexford J., Autonomous Security for Autonomous Systems, *Computer Networks* 52 (15):2908-2903, Springer, 2008.
- Kienzle, D. M., Elder, M. C., Tyree, D. and Edwards-Hewitt, J., Security Patterns Repository, Version 1.0, 2002 www.scrypt.net/~celer/securitypatterns/repository.pdf.
- Lee, KwangEui, An Automated Sensor Deployment Algorithm Based on Swarm Intelligence for Ubiquitous Environment, *International Journal of Computer Science and Network Security*, 7 (12) 2007.
- Lipson, Howard, Evolutionary Design of Secure Systems – The First Step Is Recognizing the Need for Change, Software Engineering Institute, Carnegie Mellon University, 2006-04-19; Updated 2008-10-08, 2006.
- Mahimkar, A. , Dange, J., Shmatikov, V., Vin, H. and Zhang, Y., dFence: Transparent Network-Based Denial of Service Mitigation, in Proceedings of 4th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2007), Cambridge, MA, April, 2007.
- Meszaros, G. and Doble, J., MetaPatterns: A Pattern Language for Pattern Writing, in proceedings 3rd Pattern Languages of Programming conference, 1996.
- Mouratidis, H., Giorgini, P., and Schumacher M., Security Patterns for Agent Systems, in Proceedings EuroPLoP, Irsee, Germany, June 25-39 2003.
- Microsoft Patterns & Practices Developer Center, Web Service Security, <http://msdn.microsoft.com/en-us/library/aa480545.aspx>
- Microsoft Patterns & Practices Developer Center, Perimeter Service Router Pattern, <http://msdn.microsoft.com/en-us/library/aa480606.aspx>.
- Myers, David, Play and Punishment: The Sad and Curious Case of Twixt, In Proceedings of The [Player] Conference, August 26-29, Copenhagen, Denmark, 2008.
- Parunak, H. Van Dyke, Go to the Ant: Engineering Principles from Natural Multi-Agent Systems, *Altarum Institute, Annals of Operations Research*, 75:69-101, 1997.
- Parunak, H. Van Dyke, Brueckner, Sven A. and Odell, James, Swarming Pattern Detection in Sensor and Robot Networks, in Proceedings of 10th International Conference on Robotics and Remote Systems for Hazardous Environments,

- American Nuclear Society (ANS), Gainesville, Florida, March 28-31, 2004.
- Portland Pattern Repository, <http://c2.com/ppr/index.html>.
- SOA Patterns.org, Service Perimeter Guard, www.soapatterns.org/service_perimeter_guard.asp.
- Sagarin, R. D. and Taylor, T., *Natural Security – A Darwinian Approach to a Dangerous World*, University of California Press, 2008.
- Salingaros, Nikos A., The Structure of Pattern Languages, *Architectural Research Quarterly*, 4:149-162, Cambridge University Press, 2000.
- Salustri, Filippo A., Using Pattern Languages in Design Engineering, International Conference on Engineering and Design, Melbourne August 15-18, 2005.
- Schumacher, M. and Roedig, U., Security Engineering with Patterns, in Proceedings PLoP'01, Monticello, IL, Sept. 11-15, 2001.
- Schumacher, M., Fernandez-Buglioni, E., Hybertson, D. and Buschmann, F., *Security Patterns: Integrating Security and Systems Engineering*, Wiley, 2006.
- Smathers, Douglas C. and Goldsmith, Steven Y., Agent Concept for Intelligent Distributed Coordination in the Electric Power Grid, Sandia Report SAND2000-1005, Sandia National Laboratories, 2001.
- Steel, Christopher, Nagappan, Ramesh, Lai, Ray, *Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management*, Prentice Hall, 2005.
- Wilkinson, Sophie, Plants to Bugs: Buzz Off!, Chemical and Engineering News, June 30, 2001.
- Yoder, J. and Barcalow, J. Architectural Patterns for Enabling Application Security, In Proceedings PLoP'97, Monticello, IL, Sept. 3-5, 1997.
- Yoshioka, N., H. Washizaki, and M. Katsuhisa, A Survey on Security Patterns, National Institute of Informatics, The Graduate University for Advanced Studies, Ritsumeikan University, 2008.
- Zhang, C., Zhang, J., Liu, S., and Liu, Y., Network Intrusion Active Defense Model Based on Artificial Immune System. Fourth International Conference on Natural Computation, Jinan, China, October 18-20, 2008.
- Zoccola, Mary, Homegrown, Affordable UAV Concept to be Produced for Navy Use, Naval Sea Systems Command Carderock Division, March, 2001.

Biography

Rick Dove develops and conducts research, projects, workshops, and courses for architecting and operating agile self-organizing systems. He is an adjunct professor at Stevens Institute of Technology in the School of Systems and Enterprises; and author of *Response Ability, the Language, Structure, and Culture of the Agile Enterprise*. He co-founded and chairs the INCOSE working group on System Security Engineering, and has a career history in venture start-ups, consulting, and interim executive and project management. As Principle Investigator for two successive DHS-funded projects he took massively parallel VLSI pattern recognition technology to current commercialization-in-process for security and other applications. He has a BSEE from Carnegie Mellon University.

Laura Shirey earned her BS in Computer Science from the University of Maryland in 1997, and is currently pursuing her Masters in Systems Engineering with Stevens Institute of Technology. She has developed software for several start-up companies since graduating from Maryland, and is currently the technical lead for a government software project. Her current Systems Engineering interests include agile security and risk management.