

Similarities and differences between CLASP, SDL, and Touchpoints: the activity-matrix

Koen Buyens Johan Grégoire Bart De Win
Riccardo Scandariato Wouter Joosen

Report CW 501, Oct. 2007



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

Similarities and differences between CLASP, SDL, and Touchpoints: the activity-matrix

*Koen Buyens Johan Grégoire Bart De Win
Riccardo Scandariato Wouter Joosen*

Report CW 501, Oct. 2007

Department of Computer Science, K.U.Leuven

Abstract

Development processes for software construction are common knowledge and mainstream practice in most development organizations. Unfortunately, these processes offer little support in order to meet security requirements. Over the years, research efforts have been invested in specific methodologies and techniques for secure software engineering, yet dedicated processes have been proposed only recently.

This report presents the results of a comparative study of three high-profile processes for the development of secure software, namely OWASP's CLASP, Microsoft's SDL and McGraw's Touchpoints. The key contribution of the report is a comprehensive model for comparison, the *activity-matrix*, which lists and relates all activities of the different processes and facilitates the identification of similarities and differences between the processes.

Similarities and differences between CLASP, SDL, and Touchpoints: the activity-matrix

Koen Buyens Johan Grégoire Bart De Win
Riccardo Scandariato Wouter Joosen

2007-01-08

Abstract

Development processes for software construction are common knowledge and mainstream practice in most development organizations. Unfortunately, these processes offer little support in order to meet security requirements. Over the years, research efforts have been invested in specific methodologies and techniques for secure software engineering, yet dedicated processes have been proposed only recently.

This report presents the results of a comparative study of three high-profile processes for the development of secure software, namely OWASP's CLASP, Microsoft's SDL and McGraw's Touchpoints. The key contribution of the report is a comprehensive model for comparison, the *activity-matrix*, which lists and relates all activities of the different processes and facilitates the identification of similarities and differences between the processes.

1 Introduction

The construction of secure software is still largely a matter of guidelines, best practices and undocumented expert knowledge. Current practices provide guidance for particular areas such as threat modeling, risk management, or secure coding. It is crucial for these to be combined into an integrated and more comprehensive construction method. Several advances have recently been made in the definition of processes for secure software development. However, there has been no objective comparison of these methodologies so far. Therefore, it is difficult for managers and developers to understand their strengths and weaknesses and, hence, it is hard to make an ‘informed’ decision about which one is more appropriate for the job.

The contribution of this report is the synthesis of the processes into a comprehensive model, the *activity-matrix*. This matrix lists and relates all activities of the different processes and it introduces grouping of activities into regular development phases. As such, it provides the foundations for further evaluation and comparison of the respective processes.

The report focuses on three forefront representatives, namely Microsoft’s *Security Development Lifecycle* (SDL) [2], OWASP’s *Comprehensive, Lightweight Application Security Process* (CLASP) [4] and McGraw’s *Touchpoints* [3], as they are recognized as the major players in the field. Their leading role is, among others, due to a number of characteristics that are of particular interest in the context of this study. As far as completeness is concerned, they all provide an extensive set of activities covering a broad spectrum of the development life-cycle. The processes have also undergone extensive validation. For instance, Microsoft is using SDL internally (e.g., for the Vista project). CLASP was contributed to and reviewed by several leading security companies of the OWASP consortium. The Touchpoints work has been validated over time as it has been based on experience gained from several industrial projects. Furthermore, the selected processes represent a healthy mix for comparison. SDL is considered to be more heavyweight and rigorous, making it more suitable for large organizations. CLASP, on the other hand, is lightweight and more affordable for small organizations with less strict security demands. Touchpoints is based on industrial experience gathered over several years which was subsequently distilled into an approach, making sure that the proposed activities are both executable and feasible. Finally, the selection of the processes was also influenced by the availability of thorough documentation.

The rest of this document is structured as follows: Section 2 describes the methodology that was used to define the matrix. Section 3 contains the complete activity-matrix and Section 4 briefly concludes.

2 Methodology

To begin, the team agreed on the sources to be used, since possible differences and inconsistencies may exist in different versions. For CLASP it was decided to use the documentation

of version 1.2 available at the OWASP web site.¹ For SDL the book by Howard and Lipner [2] was used and for Touchpoints the book by McGraw [3].

Based on the available sources, three by-the-book initial lists of activities were articulated. For some activities this required some interpretation of text to elicit implicit activities. Furthermore, some clean-up was necessary on these lists. This meant optimizing the hierarchical structure of activities and similar transformations. Once the lists of activities were agreed upon, they were merged by linking conceptually similar activities, which was not always straightforward, as the granularity and order of activities is not always identical and as particular activities (such as threat modeling) can have many different interpretations. Of all steps taken, this was probably the most challenging one, and the goal was to be as precise as possible.

To increase the structure and the organization of activities, the activities were subsequently organized into the different phases of a typical software development process ('Education and Awareness', 'Project Inception', 'Analysis and Requirements', 'Architectural Design', 'Detailed Design', 'Implementation', 'Testing', 'Release and Deployment', and 'Support'). Since SDL activities are already organized into stages, the activities were mapped onto the above-mentioned phases using extra information provided by MSDN [1]. As far as CLASP is concerned, activities have been assigned to phases by looking at the role that is responsible for the activity itself. Finally, Touchpoints provides a mapping of the touch points to typical software development phases, which was used accordingly.

To remove timing issues, identical activities that appeared in different phases were included only once. For instance, the SDL security push activity 'scrub attack surface' was moved to the related attack surface activities.

The result of this analysis was structured as a matrix, which contains a section for each of the aforementioned phases, with each row representing a certain activity and each column representing one of the approaches. For reasons of understandability, some (sub-phase level) grouping of activities was introduced as well.

3 The Matrix

This section presents the full matrix. The activities are conceived as a tree with every leaf node representing a concrete activity. This activity tree is mapped onto the rows of the matrix, and all concrete activities are related to the respective processes. A cell in the matrix containing a checkmark indicates that the process includes that activity. The number behind the checkmark is a reference to the original documentation of the process: it refers to the page number on which the activity is described.

¹CLASP 2 has been announced for some time now, but no new material is available from the website yet.

	SDL	CLASP	Touch - points
Education and awareness			
1.1. Baseline education			
1.1.1. Educate security basics	✓ 57	✓ 8	✗
1.2. Educate Infosec people on the applications being developed and the software development environment	✗	✗	✓ 233
1.3. Advanced education	✓ 58	✓ 8	✗
1.3.1. Plan exercises and labs	✓ 58	✗	✗
1.3.2. Track attendance and compliance	✓ 58	✗	✗
1.3.3. Measure knowledge	✓ 58	✗	✗
1.4. Share security artifacts with team	✗	✓ 8	✗
Project inception			
2.1. Determine whether the application is covered by methodology	✓ 67	✗	✗
2.2. Security team			
2.2.1. Build security leadership team	✓ 71	✗	✓ 97
2.2.2. Assign project-specific "security advisor" or "project security officer"	✓ 68	✓ 9	✗
2.2.3. Institute accountability for security issues	✗	✓ 8	✗
2.2.4. Institute rewards	✗	✓ 9	✗
2.3. Address security logistics (e.g., bugtracking tools)	✓ 72	✗	✗
2.4. Decide what types of bugs you are going to fix (determine the bug bar)	✓ 74	✗	✗
2.5. Security Metrics			
2.5.1. Identify metrics and specify their use	✗	✓ 10	✗
2.5.2. Institute data collection and reporting strategy	✗	✓ 12	✗
2.5.3. Collect and evaluate metrics (<i>ongoing during entire lifecycle</i>)	✓ 64, 129, 249	✓ 13	✓ 249
2.5.4. Improve measurement strategy	✗	✗	✓ 249
2.6. Global security policy			
2.6.1. Identify global project security policy, if necessary	✗	✓ 16	✗
2.6.2. Determine suitability of global requirements to project	✗	✓ 16	✗
2.7. Build and execute process improvement program	✗	✗	✓ 246, 250
Analysis and Requirements			
3.1. Resources and trust boundaries			
3.1.1. Identify network level design	✗	✓ 18	✗
3.1.2. Identify data-resources	✗	✓ 18	✗
3.2. User roles and resource capabilities			
3.2.1. Identify distinct resource capabilities	✗	✓ 20	✗

	SDL	CLASP	Touch - points
3.2.2.Map system roles to capabilities	x	✓ 20	x
3.3. Analysis-level threat modeling			
3.3.1.Threat agents			
3.3.1.1.Identify & describe threat agents	x	✓ 20	✓ 213
3.3.1.2.Review & revise threat agents	x	x	✓ 214
3.3.2.Use case driven analysis			
3.3.2.1.Elicit anti-requirements	x	x	✓ 215
3.3.2.2.Review & revise anti-requirements	x	x	✓ 214
3.3.2.3.Create & describe misuse cases	x	✓ 26	x
3.3.3.Resource Driven Analysis (non-system)	x	✓ 26	x
3.3.4.Attack driven analysis			
3.3.4.1.Elicit attack patterns	x	x	✓ 216
3.3.4.2.Review & revise attack patterns	x	x	✓ 214
3.3.4.3.Create & describe misuse cases	x	✓ 26	x
3.3.5.Misuse case synthesis			
3.3.5.1.Combine 3.3.2 & 3.3.4 into misuse cases	x	x	✓ 217
3.3.5.2.Review & Revise misuse cases	x	x	✓ 214
3.3.6.Misuse/abuse case ordering			
3.3.6.1.Rank misuse/abuse cases	x	x	✓ 214
3.3.6.2.Review & revise ranked misuse/abuse cases	x	x	✓ 214
3.3.7.Identify defense mechanisms for threats	x	✓ 27	x
3.3.8.Evaluate results with stakeholders	x	✓ 27	x
3.4. Security-relevant requirements			
3.4.1.Elicit legal and/or regulatory risk	x	x	✓ 155
3.4.2.Elicit financial or commercial considerations	x	x	✓ 155
3.4.3.Elicit contractual considerations	x	x	✓ 156
3.4.4.Document explicit business requirements	x	✓ 22	x
3.4.5.Develop functional security requirements	x	✓ 23	x
3.4.6.Label requirements that denote dependencies	x	✓ 24	x
3.4.7.Resolve deficiencies and conflicts between requirement sets	x	✓ 25	x
3.4.8.Determine risk mitigations for each resource	x	✓ 24	x
3.5.Execute Risk Management Framework	x	x	✓ 43,49
3.6. Requirements for the operational environment			
3.6.1.Identify requirements and assumptions for hosts	x	✓ 14	x
3.6.2.Identify requirements and assumptions for network	x	✓ 15	x
3.7. Define use scenarios for threat modeling	✓ 105	x	x
Architectural Design			
4.1.Risk assessment for 3 rd party products			
4.1.1.Gather information about reputation of COTS components and platforms	x	x	✓ 228
4.1.2.Get technology assessment from vendor	x	✓ 33	x
4.1.3.Perform security risk assessment for 3 rd party technology	✓ 94	✓ 33	x
4.2. Requirements audit			
4.2.1.Review non-security requirements	x	✓ 39	x
4.2.2.Assess completeness of security requirements	x	✓ 39	x
4.3. Architecture-level threat modeling			
4.3.1.Develop an understanding of the system	x	✓ 38	✓ 162
4.3.2.Gather a list of external dependencies	✓ 106	x	x
4.3.3.Define and validate security assumptions	✓ 106	✓ 38	x
4.3.4.Create external security notes	✓ 107	x	x

	SDL	CLASP	Touch - points
4.3.5.Create data flow diagrams	✓ 110	✗	✗
4.3.6.Identify threat types	✓ 114	✗	✗
4.3.7.Identify threats	✓ 116	✓ 39	✗
4.3.8.Perform architectural risk analysis	✗	✓ 40	✓ 163
4.3.9.Perform ambiguity analysis	✗	✓ 38	✓ 165
4.3.10.Perform weakness analysis	✗	✗	✓ 167
4.3.11.Assign risk to threats	✓ 121	✓ 40	✗
4.3.12.Identify countermeasures	✓ 124	✓ 41	✗
4.3.13.Create audit report and evaluate findings	✗	✓ 41	✗
4.3.14.Review threat models	✓ 182	✗	✗
4.3.15.Update threat model	✓ 174	✗	✗
4.4. Security design principles			
4.4.1.Refine existing application security policy	✗	✓ 30	✗
4.4.2.Determine implementation strategy for security services	✗	✓ 30	✗
4.4.3.Build hardened protocol interfaces	✗	✓ 31	✗
4.4.4.Design hardened application interfaces	✗	✓ 32	✗
Detailed Design			
5.1. Assess the privacy impact rating of the project	✓ 96	✗	✗
5.2. Software attack surface reduction			
5.2.1.Remove unimportant features	✓ 78	✗	✗
5.2.2.Determine who needs access from where	✓ 78	✗	✗
5.2.3.Reduce privileges	✓ 78	✗	✗
5.2.4.Identify system entry points	✗	✓ 28	✗
5.2.5.Map roles to entry points	✗	✓ 28	✗
5.2.6.Map resources to entry points	✗	✓ 28	✗
5.2.7.Scrub attack-surface	✓ 175	✗	✗
5.3. Class design annotation			
5.3.1.Map data elements to resources and capabilities	✗	✓ 35	✗
5.3.2.Annotate fields with policy information	✗	✓ 35	✗
5.3.3.Annotate methods with policy data	✗	✓ 35	✗
5.4. Database security configuration			
5.4.1.Identify candidate configuration	✗	✓ 37	✗
5.4.2.Validate configuration	✗	✓ 37	✗
5.5. Make your product updatable	✓ 211	✗	✗
Implementation			
6.1. Security analysis tools for source management process			
6.1.1.Select analysis technology or technologies	✓ 145	✓ 42	✗
6.1.2.Determine analysis integration point	✗	✓ 42	✗
6.1.3.Integrate analysis technology	✗	✓ 42	✗
6.2. Ambiguity analysis of specification	✗	✓ 45	✗
6.3. Perform automated source-level security review	✓ 145	✓ 48	✓ 105
6.4. Perform manual code inspection	✓ 172	✗	✗
6.5. Implement security	✗	✓ 45	✗
6.5.1.Use coding guidelines during implementation	✓ 148	✓ 45	✗
6.5.2.Implement specification	✗	✓ 45	✗
6.5.3.Implement interface contracts	✗	✓ 44	✗
6.6. Create configuration tools for end-users	✓ 139	✗	✗
6.7. Prepare documentation			
6.7.1.Write user and help manuals	✓ 136	✗	✗
6.7.2.Write administrator manuals	✓ 136	✗	✗
6.7.3.Write developer documentation	✓ 136	✗	✗

	SDL	CLASP	Touch - points
6.8. Operational security guide			
6.8.1. Document pre-install configuration requirements	x	✓ 54	x
6.8.2. Document application activity	x	✓ 54	x
6.8.3. Document the security architecture	x	✓ 54	x
6.8.4. Document security configuration mechanisms	x	✓ 54	x
6.8.5. Document significant risk and compensating controls	x	✓ 54	x
6.9. Addressing reported security issues (implementation time)			
6.9.1. Assign issue to investigator	x	✓ 46	x
6.9.2. Assign likely exposure and impact	x	✓ 46	x
6.9.3. Determine and execute remediation strategies	x	✓ 46	x
6.9.4. Validate remediation	x	✓ 46	x
7.1. Security testing			
7.1.1. Perform risk based security testing	x	x	✓ 187
7.1.2. Identify and test functional security requirements	x	✓ 50	✓ 193
7.1.3. Identify resource-driven security tests	x	✓ 50	x
7.1.4. Perform adversarial security testing	x	x	✓ 193
7.1.5. Perform unit testing	x	x	✓ 188
7.1.6. Perform system testing	x	x	✓ 188
7.1.7. Build tests around risks in the system	x	x	✓ 203
7.1.8. Test states and state preservation	x	x	✓ 203
7.1.9. Test for race conditions	x	x	✓ 203
7.1.10. Perform fuzz testing	✓ 153	x	✓ 201 179
7.1.11. Perform penetration testing	✓ 164	x	✓ 171
7.1.12. Identify other relevant security tests	x	✓ 50	x
7.1.13. Execute security tests	x	✓ 51	x
7.1.14. Implement test plan	x	✓ 51	x
7.1.15. Perform run-time verification	✓ 165	x	x
7.1.16. Verify security attributes of resources	x	✓ 34	x
7.1.17. Receive permission to perform security testing on 3 rd party components	x	✓ 34	x
7.1.18. Perform security testing on 3 rd party components	x	✓ 34	x
7.2. Security push			
7.2.1. Prepare for the security push	✓ 170	x	x
7.2.2. Scrub documentation	✓ 176	x	x
7.3. Final security review			
7.3.1. Coordinate product Team	✓ 182	x	x
7.3.2. Review unfixed security bugs	✓ 183	x	x
7.3.3. Validate correct use of tools	✓ 184	x	x
Release & Deployment			
8.1. Code sign-off	✓ 215	x	x
8.2. Upload debugging symbols to central server	✓ 215	x	x
8.3. Code signing			
8.3.1. Obtain code signing credentials	x	✓ 53	x
8.3.2. Identify signing targets	x	✓ 53	x
8.3.3. Sign identified targets	x	✓ 53	x
8.4. Fine-tune access controls (network & OS) and configure the monitoring and logging	x	x	✓ 232
Support			
9.1. Security response planning			
9.1.1. Build a security response center	✓ 191	x	x

	SDL	CLASP	Touch - points
9.1.2. Provide means of communication for security issues	x	✓ 55	x
9.1.3. Create your response team	✓ 208	x	x
9.1.4. Find the vulnerabilities before researchers do	✓ 212	x	x
9.2. Addressing deployment-time security issues (Security response execution)			
9.2.1. Decide on what to skip	✓ 221	x	x
9.2.2. Vulnerability reporting & acknowledgment	✓ 195	✓ 56	x
9.2.3. Determine vulnerability impact (triaging)	✓ 195	x	x
9.2.4. Manage relationship with bug reporter	✓ 199	✓ 56	x
9.2.5. Create security bulletin/advisory (Content creation)	✓ 202	x	x
9.2.6. Create the fix	✓ 197	✓ 56	x
9.2.7. Test the fix	✓ 201	x	x
9.2.8. Release the fix and the bulletin/advisory	✓ 203	✓ 57	x
9.2.9. Improve process based on lessons learned	✓ 205	x	x

4 Conclusion

In this report three secure software development processes, namely OWASP's CLASP, Microsoft's SDL and McGraw's Touchpoints are summarized into a comprehensive model. The report has articulated and structured the processes into activities, and *demythified the relationships* between the different processes. The complexity of this should not be underestimated, as illustrated, for instance, by the overlap among the different techniques used during the elicitation of requirements and threats. This matrix can, and has been the basis for the application, further study and comparison of secure development processes. It would be useful to extend the matrix with other processes if sufficient information and in-depth description is available.

References

- [1] Msdn: Security development lifecycle phases, 2005. <http://msdn2.microsoft.com/en-us/library/ms995349.aspx>.
- [2] M. Howard and S. Lipner. *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably More Secure Software*. Microsoft Press, 2006.
- [3] G. McGraw. *Software Security: Building Security In*. Addison Wesley, 2006.
- [4] OWASP. Comprehensive, lightweight application security process. <http://www.owasp.org>, 2006.