Association for Information Systems
AIS Electronic Library (AISeL)

CONF-IRM 2017 Proceedings

International Conference on Information Resources Management (CONF-IRM)

5-1-2017

Security V&V Within Software SMEs: A Socio-Technical Interaction Network Analysis

Matthew Nicolas Kreeger
Royal Holloway, University of London, matthew.kreeger@thales-esecurity.com

G Harindranath
University of London, g.harindranath@rhul.ac.uk

Follow this and additional works at: http://aisel.aisnet.org/confirm2017

Recommended Citation
Kreeger, Matthew Nicolas and Harindranath, G, "Security V&V Within Software SMEs: A Socio-Technical Interaction Network Analysis" (2017). *CONF-IRM 2017 Proceedings*. 28.
http://aisel.aisnet.org/confirm2017/28

This material is brought to you by the International Conference on Information Resources Management (CONF-IRM) at AIS Electronic Library (AISeL). It has been accepted for inclusion in CONF-IRM 2017 Proceedings by an authorized administrator of AIS Electronic Library (AISeL). For more information, please contact elibrary@aisnet.org.

SECURITY V&V WITHIN SOFTWARE SMES: A SOCIO-TECHNICAL INTERACTION NETWORK ANALYSIS

Matthew Nicolas Kreeger
Royal Holloway, University of London
and Thales e-Security
matthew.kreeger@thales-esecurity.com

G. Harindranath
Royal Holloway, University of London
G.Harindranath@rhul.ac.uk

Abstract:

Within this paper we provide insight into how the activities associated with security verification and validation (V&V) are practiced, supported, and perceived, within software SMEs. We justify the importance of studying security V&V as a socio-technical activity and employ the Socio-Technical Interaction Network (STIN) framework when presenting the results of an industry-based empirical study. In summary, the results indicate that software SMEs are significantly less confident in their engagement with security-focused V&V activities as opposed to traditional software V&V. This includes their ability to perform and own the activities, as well as how they are supported and managed within the organisations studied. This suggests that security-focused V&V activities have not reached the same degree of maturity as the more traditional software V&V activities within software SMEs.

Keywords:

Security Verification and Validation, Software SMEs, Socio-Technical Interaction Networks

1. Introduction: Motivation and Study Context

The reliance on software for our everyday existence is now without question, however, this reliance requires a dependability that is not universally evident in the software currently being developed and deployed. Developing software, regardless of its intended application, is an inherently complex task. However, software failures, and software project failures, lead to costs being incurred by both the software consumer and producer - costs which could potentially be mitigated. It is reported that greater than 50% of all released software contains defects that affect its execution in some form (Shull et al., 2002) and, although defects are, in general, costly to both users and organisations, vulnerabilities are regarded as having greater impact (McGraw, 2006). The software producer can lose revenue or go out of business (Wysopal et al., 2006). Further, it is apparent that many vulnerabilities are the result of a small number of causes - implying that the same mistakes continue to be made (Piessens, 2002). The focus on SMEs is pertinent when we consider that they are fundamentally important to the majority of the world's economies (Mac an Bhaird, 2010). Also, most software producing organisations are small (Fayad et al., 2000). However, despite the growth and contribution of the software industry to national economies, there are many examples of software quality lapses - thus impacting the view that software can be used to build secure systems (Parnas and Lawford, 2003).

In addition, we observe that the software industry spends more money on locating and addressing defects than any other activity. Software verification and validation (V&V) is known to comprise a substantial share of a project's budget e.g. it is typical for software testing to account for at least 50% of a project's costs (Myers, 2004). More time is also being spent on such activities (Sung and Paynter, 2006). Therefore, it is clearly concerning that considerable financial outlay and human effort is being expended on activities tasked with

locating defects but yet, as noted above, we continue to release software with defects - often making the same mistakes. It is no wonder then that vulnerabilities continue to be reported (Eschelbeck, 2005). This is a clear indication that there must be a problem somewhere with actual security V&V practice. However, as software is constructed by humans and, therefore, is not going to be perfect (Patton, 2005), we further observe that it is a problem that cannot be solved purely by adopting a technical perspective. We note that the majority of V&V literature primarily focuses upon technical aspects e.g. studies comparing specific methods or tools. It is important to consider software V&V as a socio-technical activity.

2. Software V&V: A Socio-Technical Activity

Software V&V involves the interaction between people (e.g. engineers and managers) and their interactions with specific technical methods and technologies so as to perform their allocated tasks. Socio-technical issues (such as those within Table 1) are known to influence software V&V, for example: engineer attitudes can affect unit testing within an organisation (Ellims et al., 2004); developer motivation needs improving in terms of unit testing (Runeson, 2006); and motivation to review another's work is a recognised challenge (Kollanus and Koskinen, 2006). Significantly, we note that such socio-technical issues "affect everything about software engineering, including the adoption and implementation of software engineering processes and technologies" (Henry, 2005, p. 110). It is, therefore, important to understand both the social and the technical aspects of security V&V if we are to improve existing practice by means of process improvement and tool adoption. Further, we observe that such social factors have real impact on the quality of the software being developed by software producing organisations (Bird et al., 2009), with many software failures being explained by human factors (Tomayko and Hazzan, 2004). As such, we conduct our study through use of the Socio-Technical Interaction Network (STIN) framework.

Interaction	As software products are generally too complex to be developed by individuals, software testing requires tools that support team communication (Mosley and Posey, 2002). Software reviews also require effective teamwork (Aurum et al., 2002) and, specific security V&V techniques, such as fuzz testing, are known to be a combined effort "between security people and quality assurance people" (Takanen et al., 2008, p. xx).
Collaboration and cooperation	Test engineers interact with different people (both technical and non-technical), at different levels, internal and external to their organisation (Hass, 2008). Software reviews should be conducted with a cooperative team spirit (Jawadekar, 2004) and test engineers "must find a way to focus on the software defect without seeking to place blame" (Everett and McLeod, Jr., 2007, p. 20). Conflict can lead to reluctance in developer and test engineer cooperation (Vogel, 2011).
Motivation	Software V&V involves both creative and intellectually challenging tasks (Myers, 2004), however, a lack of motivation can impact these activities e.g. (Runeson, 2006; Kollanus and Koskinen, 2006). Software engineers are known to be motivated by conflicting needs (Fairley, 2009), however, software testers are known to be motivated by an interest in the task (it must be viewed as an intellectual challenge), as well as by reward and praise (Hass, 2008). A lack of management support is known to cause software testers to lose both motivation and interest (Perry, 2006).

Table 1: The social and human aspects of software V&V

2.1. Socio-Technical Interaction Networks (STINs)

A STIN is defined as a "network that includes people (including organizations), equipment, data, diverse resources (money, skill, status), documents and messages, legal arrangements and enforcement mechanisms, and resource flows" (Kling et al., 2003, p. 48). Its use highlights the relationships between people and their relationships with technologies (Walker, 2008). We have already observed that software V&V involves interaction between people, as

well as their interactions with specific technical methods and technologies in order to perform their tasks. In summary, STIN identifies groups of interactors and then models the relationships between them (van der Merwe, 2010). Kling et al. distil the methodology for constructing a STIN into a series of steps as captured in Table 2.

Step 1: Identify a relevant population of system interactors and core interactor groups	This step is concerned with identifying and characterising potential interactors i.e. identifying the actors, their roles and the way they participate within the system. The identified interactors are then grouped together by role. Kling et al. also note that the identified groups might overlap and, in some instances, have conflicting roles (Kling et al., 2003).
Step 2: Identify incentives	Incentive structures are then identified. This involves understanding the potential motivations of the interactors (Meyer, 2007).
Step 3: Identify excluded actors and undesired interactions	It is critical to not only understand desired interactions, but also undesired ones (Kling et al., 2003). This is deemed an often overlooked step in socio-technical research (Meyer, 2007).
Step 4: Identify existing communication forums	Within this step the existing communication systems, used by the interactors, are identified.
Step 5: Identify resource flows	Involves understanding how resources flow throughout the network. Resource flows can have direct and indirect influence on interactions within the network and involves "following the money" (Kling et al., 2003). Resource flows also include expertise (Meyer, 2007).

Table 2: Analysis steps of a Socio-Technical Interaction Network

STIN has been used to study, and explain, a number of different complex social systems involving information technology (Urquhart and Currell, 2010). For example, Scacchi applied STIN to study free / open source software development processes to explore the social and technological interactions found within different research and development communities (Scacchi, 2005). Specifically, Scacchi acknowledges that STIN draws attention to the "relationships that interlink what people do in the course of their system development work to the resources they engage and to the products (software components, development artifacts, and documents) they create, manipulate, and sustain". Scacchi concludes that STINs provide a "better" way to "observe the contexts in which people carry out software development processes and related work practices". As noted by Kling et al.: "STIN models help us to understand human behaviors in the use of technology-mediated social settings" (Kling et al., 2003, p. 48). The activity of security V&V fits perfectly within the definition of a technology-mediated social setting.

3. Research Methods

We devised an instrument which comprises between 30 and 62 open and closed questions (the number being dependent on a respondent's responses). As the state of security V&V, within software SMEs, is effectively an empirical unknown (Kreeger and Harindranath, 2012), a survey is a suitable means of obtaining a snapshot of the current status (Wohlin et al., 2003). Importantly, it also permits the beliefs and opinions of a sample to be generalised (Easterbrook et al., 2008). We pre-tested the instrument with 11 subject matter experts prior to circulation. In summary, just over 15 hours were spent sitting with the respondents, resulting in instrument refinement.

Our sample was derived through use of the Financial Analysis Made Easy (FAME) database (Bureau van Dijk, 2017), which is a well-established means of determining a study sample. Our search strategy applied the following restrictions: the organisation must be a software producer i.e. fall within SIC sub-class 62.01/1 or 62.01/2 and an SME i.e. not exceeding 249

employees and a turnover of €50 million (we did not utilise the "use estimates" option available within FAME). In addition to the devised search strategy, we also applied some further refinement to the returned set of results. Employing an additional filtering step is consistent with others utilising FAME e.g. (Martínez-Caro and Cegarra-Navarro, 2010). This resulted in a total of 160 organisations being identified. We further supplemented this probability sample through convenience sampling.

3.1. Response Rate Analysis: Subject Sensitivity

It is observed that "[t]he topic of a survey has a clear impact on people's willingness to take part in it" (Tourangeau et al., 2000, p. 261). Certainly, topic sensitivity is considered one of the primary factors resulting in survey non-response (Kays et al., 2013). As our research focuses on information security, we recognise that this is "one of the most intrusive types of organization research" (Kotulic and Clark, 2004, p. 604). However, it is with this in mind that we acknowledge that whilst our survey response rate is 18.13% (in terms of the FAME sample), we feel, like Kotulic and Clark, that this can be explained by virtue of our study falling into both an under-researched area, as well as one that covers a sensitive topic. Based on this, we adopted convenience sampling to further supplement the responses received. This resulted in an additional 8 responses, and thus a total of 37 responses overall. By way of comparison, we observe that Thörn and Gustafsson consider the 27 responses received to their survey - which also focused on software development within SMEs - as "a response rate that is comparable to other surveys in software engineering" (Thörn and Gustafsson, 2008, p. 23). Singer et al. found "a consistent response rate of 5% to software engineering surveys" (Singer et al., 2008, p. 15) and a response rate of 12.6% within one SME study was considered: "within reasonable and expected limits" (Rasmussen and Thimm, 2009, p. 86). We, therefore, consider the level of response to our survey as being acceptable. This view is further strengthened when considering the type of respondents reached, namely: experienced, senior employees situated within a technology-focused, SME organisational context.

3.2. Respondent Demographics

Approximately three quarters of the respondents were from medium-sized organisations (<250 employees), the remainder being considered small (<50 employees). The organisations vary in age from 1 - 2 years to over 20 years, however, the majority are classified as being "mature" (24% indicating 6 - 10 years) or "old" (67% indicating over 10 years), which adds strength when interpreting the results. Specifically, if the majority of organisations were considered "entrants" (<1 year old) it would be easy to dismiss any findings as being unreliable. Further, the organisations target a variety of industry sectors with their products. We find that 10 industry sectors are represented e.g. information and communication (including products such as secure collaboration and network management solutions), financial and insurance activities (including online trading systems) and the education sector (including parent to school communication systems). This diversity helps to provide further insight and ensures that any results can be generalised. We also observe that the organisations utilise a variety of software development methodologies, which clearly influences the V&V being performed. However, our results support the "increasing popularity of agile methods" (Ryan and O'Connor, 2013, p. 1615) as we find that Agile is practiced in some form by 86% of the respondents. Test-driven development is the next most widely employed methodology (41%), followed by Waterfall (35%) and then the V-Model (32%). Seventy percent of the organisations employ at least two different methodologies.

The respondents should be well positioned to ensure that the data received is an accurate reflection of the situation. As such, 81% are leads, managers or senior managers, holding positions such as CTO, COO, Head of Software, Director of Software Development, Test Manager, Senior QA Manager or Team Leader. Tenure is also an important factor to consider when interpreting the responses as it "helps define an employee's knowledge and perspective on the organization" (Crawford and Leonard, 2012, p. 64). Over a third of the respondents have worked for their current organisation between 3 - 5 years. Approximately a quarter exceeding 6 years, the remainder under 3. The average tenure for a software engineer is under two years (Appelbaum, 2001). However, it is also important to consider a respondent's software experience as a whole as such collective experience will result in a more critical eye when reflecting upon their current situation. Approximately 95% of the respondents have at least 6 - 10 years of software experience, with 62% having over 11 years. All respondents have at least 3 - 5 years of software experience. Seventy percent of the respondents have previously held test-related roles, 30% development-based roles and 22% a variety of other software roles (e.g. support engineer and consultant). Approximately 65% of the respondents have worked for other SMEs, 81% for large organisations. In summary, we believe that the respondents engaged bring both a wealth of experience and a variety of perspectives.

4. Results

We begin our analysis by determining whether the various software and security V&V activities are performed within the surveyed organisations and how frequently. As captured within Figures 1 and 2, it is apparent that the traditional software V&V activities are performed with more regularity than the security-focused V&V activities. Aside from the drop in regularity that an activity is performed, we also see a marked increase in the number of "Do Not Know" responses. We now proceed to structure our analysis of software and security V&V practice within SMEs through the application of STIN.

4.1. Step 1: Interactor Analysis

We find a fairly diverse set of human interactors involved with the various software and security V&V activities. Unsurprisingly, developers and test engineers are the primary (i.e. more frequently involved) type of interactor, with the traditional role divide between these interactor groups being evident - namely: developers are much more involved in performing and owning review-based activities and test engineers the test-based activities. All of the organisations surveyed clearly distinguish between these two interactors which, given the wide adoption of Agile (i.e. advocating the "generalist") was a little surprising.

Secondary interactors include outsourced engineers and customers. We found a greater dependency on outsourcing for the security V&V activities - with a desire to further increase its use - both in terms of performing, and owning, the security-focused activities. Whilst there is also an increase in view (but not as significant) that the software V&V activities should be outsourced, this was in terms of performing an activity and not its ownership. There was also a desire to further engage customers across both sets of V&V activities. Lesser involved interactors include business analysts, product management, support and sales teams. There were also notable "exceptions", with one organisation involving their internal infrastructure team "*to advise on the type of security issues we might hit*", and another involving their Chief Security Officer. However, in both cases, room for improvement was evident. For example, the Chief Security Officer was not involved in any security design reviews and the internal infrastructure team was noted as needing to become engaged with the developers and test engineers performing penetrating testing.

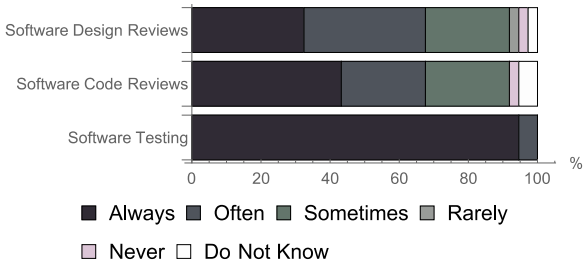

Figure 1: Software V&V activities

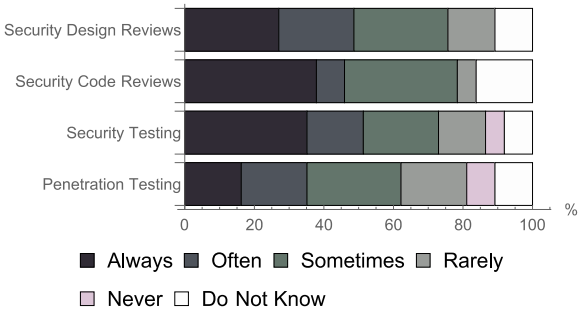

Figure 2: Security V&V activities

We also observe diversity in terms of the non-human interactors. Specifically, we found that the software V&V activities were more likely to have tool support, with the respondents more readily able to discuss the tools used within the context of such activities. When it came to discussing the tools supporting the security-focused activities the respondents were more likely to indicate "Do Not Know". Two respondents indicated this was due to a reliance on outsourcing which, with the noted desire to increase its use, the situation can only worsen. In a similar vein, we find that the software V&V activities were more likely to have a governing process than the corresponding security-focused activities (the respondents were also more confident in articulating the situation).

4.2. Step 2: Incentives

A variety of incentives were found to help motivate the identified interactors in performing the V&V activities. Specifically, we found the primary motivator in performing any V&V activity was through a desire to improve software quality. However, we found that this to be more prevalent in terms of the software V&V activities than the corresponding security-focused activities. We ascribe this to interactors not affording due importance to security as a quality attribute. The second most frequently indicated motivator was acknowledging and adhering to a governing process. We found that this was more pronounced in terms of the software V&V activities. This is further evidence that such activities have reached a higher level of maturity. There was very little indication to suggest that the interactors are motivated by a sense of fun when performing any of the V&V activities. Conversely, there is little indication that they are motivated to perform the activities to solely "tick a box".

Collectively what these results show, aside from a desire to improve software quality, is that the interactors appear to be more motivated to perform the V&V activities due to external influences (i.e. a mandating process, instruction from management and requests from customers) than by intrinsic factors - such as deriving enjoyment from the activity itself. That we found the most prevalent incentive to be a desire to improve software quality is supported by the existing literature on motivation in software engineering e.g. (Hall et al., 2008). However, we confirm this within SMEs and within a V&V context. We also found that this level of intrinsic motivation varies dependent on the type of activity. The software V&V activities were more likely to be motivated by this factor than security-focused activities. This is clear indication that security - as a quality attribute - is not afforded, or perceived by the interactors as having, as much worth as other quality attributes.

4.3. Step 3: Excluded Actors and Undesired Interactions

It is apparent that undesired interactions - both of a social and technical nature - exist, as well as that some of the identified interactors either are - or should be - excluded from certain activities. Specifically, the activities, as practiced within the vast majority of organisations surveyed, suffer due to resourcing constraints and / or social factors. Whilst we find commonality in some of the reported constraints and factors, there are also marked differences. For example, we find that the traditional activities are more likely to be impacted by a lack of people, whereas the security-focused activities are more likely to be impacted by a lack of training and knowledge (this is the most common reason given for never performing a security V&V activity). In addition, we find that the security-focused activities are less likely to receive management support, whereas the traditional activities are more likely to suffer from poor communication.

We find that test engineers are more likely to be excluded from review-based activities and developers from test-based activities. This helps further emphasise the split between interactor type and activity. However, we find an acknowledgement, within the organisations, to move away from such separation. We speculate the significant adoption of Agile has helped foster this view. Although, overall, the most excluded interactor was the test engineer - which helps emphasise the imbalances which exist between the two primary interactors. There are more reported instances of actively wanting to exclude interactors - in terms of the security-focused activities - than the traditional activities.

The security-focused activities were also more likely to suffer from a lack of interactor motivation. This is somewhat unsurprising, since as the security-focused activities were more likely to be impacted by a lack of management support, it is, by extension, probable, that the level of motivation exhibited by the interactors will be further weakened. In turn, it is this lack of motivation which is likely to result in the activities being performed less effectively. This is liable to result in a vicious cycle, whereupon, the organisation is not witness to any significant benefits - as the activities are being practiced in a sub-optimal manner - therefore, there is not such a strong incentive to invest and actively support the activities. We find evidence to support the view that only by failing in terms of software security is an organisation more likely to become actively invested in such activities. As stated by one respondent: "*I think security only raises its head when things go wrong*", with another indicating: "*I would say we generally have the (in my opinion, wrong) opinion that if customers want to test our products for security issues they can, and we generally only perform security testing when forced. I am attempting to change the culture within the organisation to bring the importance of security testing up the scale. We have had some incidents lately with customers that are starting to support my case*". As the costs of such failure can be high, this is an expensive lesson to learn.

4.4. Step 4: Existing Communication Forums

Although we identified the developer and test engineer as the primary interactors, we find, when it comes to communicating the results of the V&V activities, that it is the developer who is the most informed type of interactor. For example, we find that the results of review-based activities are not typically communicated to the test engineer. This not only highlights the traditional divide between the two interactors, but also shows an undesired state, since: "[c]ode reviews allow for dissemination of key information" (Coram and Bohner, 2005, p. 365). As Coram and Bohner observe, code reviews, as a means of communication, permit familiarisation with a product's "inner workings". By not possessing this level of

understanding i.e. treating it as a black-box, the ability to detect vulnerabilities is reduced (Potter and McGraw, 2004). Therefore, whilst this does not impact traditional test-based activities, it does impact those which are security-focused.

We also find that test-based activities - both software and security-focused - are more likely to communicate results through formal documentation than review-based activities. This is probably in part due to the levels of outsourcing found in terms of penetration testing, as well as the fact that the test tools reported as being in use typically offer the ability to auto-generate sets of results (which is supported by one respondent indicating that the "*[r]esults of automated testing are a factor in all releases*"). Further, a combination of communication mechanisms are used per activity e.g. formal documentation and verbal communication.

4.5. Step 5: Resource Flows

The average level of expenditure, per activity, is found to be higher in terms of the traditional activities than the corresponding security-focused activities. As captured by one respondent, the impact of a "*[l]ack of resources*" is the reason why "*[n]o aspect of security is tested*". Software testing incurs the most costs of any V&V activity (averaging 37% on a typical project) - a finding which echoes existing literature - but one which we confirm within an SME context. The respondents were more likely to indicate that additional investment was required in terms of the security-focused activities than the traditional activities. Whilst this is a clear indication that the respondents acknowledge the value of such activities, it is through analysis of the results that we find such investment needs to be carefully directed, for example:

- Whilst a lack of budget is the primary reason for why tools are not used, it is the primary reason for why both traditional and security-focused tools are not used.
- Tools supporting communication between the interactors can, and are, applied to both the traditional and the security-focused activities.
- The average levels of automation (we recall that automation is considered expensive (Everett and McLeod, Jr., 2007)), per activity, are comparable when contrasting the traditional and corresponding security-focused activities.

However, where investment could make a significant difference is in terms of training and improving the level of employee knowledge within the organisations as we find the level of training afforded to the respondents as being rather limited. Specifically, there was negligible review-based related training found but, in terms of software testing, there was considerably more (approximately 70% of the organisations provided such training). However, this level of support drops significantly for security-focused test-based training (falling to ~30% for security testing and ~20% for penetration testing). We find that the majority of training received was over 12 months ago. We also find that organisations are, overall, more likely to support training in forms which incur the least direct costs e.g. reading and mentoring (which, collectively, comprise almost 63% of all training received). In summary, we find that the levels of training afforded by the organisations are constrained - being both limited in terms of opportunity and frequency, as well as being potentially out-of-date.

5. Conclusions

The impact of vulnerable software is well known and far reaching in its consequences: information insecurity incurs significant costs (reportedly in the billions (Schneier, 2007)). Coupled with the observation that software SMEs are developing significant products (Fayad

et al., 2000), it is clearly essential that such organisations ensure the security of their products through a rigorously applied, and well supported, security V&V process. However, our results show that within software SMEs, security V&V is performed less frequently than the traditional software V&V activities; also, we find that the respondents had more difficulty in elaborating upon the security-focused activities being performed within their organisations. We thus conclude that software SMEs are significantly less active, and confident, in their engagement with security V&V as opposed to software V&V. Specifically, they have greater confidence in their ability to perform, and own, the software V&V activities. We also find that these activities are better supported and managed. This is confirmed by:

- Very limited engagement with internal security expertise outside of the core engineering team e.g. Chief Security Officer or infrastructure team. This suggests that software SMEs are not utilising their internal resources as effectively as possible.
- A noticeable absence of internal roles dedicated to performing security V&V related activities e.g. security test engineer or penetration tester (or even a more generic security engineer). This is likely to result in security taking second place to other quality attributes and / or the increased reliance on outsourcing.
- The traditional software V&V activities being more likely to have supporting tools and a governing process. This shows that the security-focused V&V activities are not as well supported and managed which, in turn, further helps deprioritise security as a quality attribute.

Collectively, the above highlights that the security-focused V&V activities have not reached the same degree of maturity as the more traditional software V&V activities within software SMEs. However, the finding that a lack of security-focused training and knowledge is one of the primary factors impacting the security-focused activities further emphasises, and supports, the view that security V&V encompasses a complex set of activities, requiring a blending of knowledge, across a variety of domains. As found, this has a greater detrimental impact on the security-focused activities. Unfortunately, whilst we find that there is a recognised need for further investment in these activities, it is clear that this is yet to result in the provision of training within the SMEs. In general, training was found to be rather limited, and whilst this is supported by the existing literature regarding SMEs, we are able to confirm that this extends to encompass all software V&V activities. Further compounding the situation, we find that the security-focused activities are less likely to receive management support and are more likely to suffer through a lack of interactor motivation. In conclusion, the findings suggest that software SMEs need to improve on how security V&V is practiced, supported, and perceived, in order to avoid releasing software products containing vulnerabilities which could then be exploited.

It was by identifying the human and non-human interactors, and exploring their relationships, that a more nuanced understanding was developed. For example, whilst some V&V studies only enumerate the tools used (effectively, just performing interactor analysis), we examined the relationships between interactors (i.e. STIN steps 3 - 5) to show how the tools were used and what impacted their use. Equally, if we had just focused on network relationship analysis, the results may have been misinterpreted. For example, we found penetration testing to be the activity least impacted by a lack of time (which was, overall, the most frequently reported constraint across all of the activities). However, the interactor analysis enabled us to identify that a significant reliance on outsourced engineers existed for this activity i.e. any time constraints caused by limited resourcing (a common problem within small organisations)

would be effectively removed. Therefore, we found that the adoption of STIN permitted a more complete and thus, accurate understanding of security V&V within SMEs.

References

Appelbaum, E. (2001). Transformation of work and employment and new insecurities. *The Future of Work, Employment and Social Protection: The search for new securities in a world of growing uncertainties*, 17-37. International Institute for Labour Studies.

Aurum, A., et al. (2002). State-of-the-Art: Software Inspections after 25 Years. *Software Testing, Verification and Reliability*, 12(3), 133-154.

Bird, C., et al. (2009). Putting it All Together: Using Socio-Technical Networks to Predict Failures. *ISSRE '09: Proceedings of the 20th International Symposium on Software Reliability Engineering*, 109-119.

Bureau van Dijk. (2017). *FAME: The definitive source of information on companies in the UK and Ireland*. http://www.bvdinfo.com/en-gb/our-products/company-information/national-products/fame. Retrieved: 28th of March, 2017.

Coram, M & Bohner, S. (2005). The Impact of Agile Methods on Software Project Management. *ECBS '05: Proceedings of 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, 363-370.

Crawford, J. & Leonard, L. N. K. (2012). Predicting post-meeting work activity in software development projects. *Team Performance Management: An International Journal*, 18(1/2), 59-77.

Easterbrook, S., et al. (2008). Selecting Empirical Methods for Software Engineering Research. *Guide to Advanced Empirical Software Engineering*, 285-311. Springer.

Ellims, M., et al. (2004). Unit Testing in Practice. *ISSRE '04: Proceedings of the 15th International Symposium on Software Reliability Engineering*, 3-13.

Eschelbeck, G. (2005). The Laws of Vulnerabilities: Which security vulnerabilities really matter? *Information Security Technical Report*, 10(4), 213-219.

Everett, G. D. & McLeod, Jr., R. (2007). *Software Testing: Testing Across the Entire Software Development Life Cycle*. IEEE Press / John Wiley & Sons.

Fairley, R. E. (2009). *Managing and Leading Software Projects*. John Wiley & Sons.

Fayad, M., et al. (2000). Software Engineering in the Small. *Communications of the ACM*, 43(3), 115-118.

Hall, T., et al. (2008). What Do We Know about Developer Motivation? *IEEE Software*, 25(4), 92-94.

Hass, A. M. (2008). *Guide to Advanced Software Testing*. Artech House.

Henry, T. R. (2005). Software Development Productivity: Considering the Socio-Technical Side of Software Development. *Issues in Information Systems*, 6(2), 110-116.

Jawadekar, W. S. (2004). *Software Engineering: Principles and Practice*. McGraw-Hill.

Kays, K. M., et al. (2013). Best Practice in Online Survey Research with Sensitive Topics. *Advancing Research Methods with New Technologies*, 157-168. IGI Global.

Kling, R., et al. (2003). A Bit More to It: Scholarly Communication Forums as Socio-Technical Interaction Networks. *Journal of the American Society for Information Science and Technology*, 54(1), 47-67.

Kollanus, S. & Koskinen, J. (2006). Software Inspections in Practice: Six Case Studies. *Product-Focused Software Process Improvement*, 377-382. Springer.

Kotulic, A. G. & Clark, J. G. (2004). Why there aren't more information security research studies. *Information & Management*, 41(5), 597-607.

Kreeger, M. N. & Harindranath, G. (2012). Security Verification and Validation by Software SMEs: Theory versus Practice. *Conf-IRM '12: Proceedings of the 2012 International Conference on Information Resources Management*.

Mac an Bhaird, C. (2010). *Resourcing Small and Medium Sized Enterprises: A Financial Growth Life Cycle Approach*. Springer.

Martínez-Caro, E. & Cegarra-Navarro, J. G. (2010). The impact of e-business on capital productivity: An analysis of the UK telecommunications sector. *International Journal of Operations & Production Management*, 30(5), 488-507.

McGraw, G. (2006). *Software Security: Building Security In*. Addison-Wesley.

Meyer, E. T. (2007). *Socio-Technical Perspectives on Digital Photography: Scientific Digital Photography Use by Marine Mammal Researchers*. Indiana University.

Mosley, D. J. & Posey, B. A. (2002). *Just Enough Software Test Automation*. Prentice Hall.

Myers, G. J., et al. (2004). *The Art of Software Testing*. Second Edition. John Wiley & Sons.

Parnas, D. L. & Lawford, M. (2003). Inspection's Role in Software Quality Assurance. *IEEE Software*, 20(4), 16-20.

Patton, R. (2005). *Software Testing*. Second Edition. Sams Publishing.

Perry, W. E. (2006). *Effective Methods for Software Testing*. Third Edition. John Wiley & Sons.

Piessens, F. (2002). A Taxonomy of Causes of Software Vulnerabilities in Internet Software. *Supplementary Proceedings of the 13th International Symposium on Software Reliability Engineering*, 47-52.

Potter, B. & McGraw, G. (2004). Software Security Testing. *IEEE Security & Privacy*, 2(5), 81-85.

Rasmussen, K. B. & Thimm, H. (2009). Fact-Based Understanding of Business Survey Non-Response. *The Electronic Journal of Business Research Methods*, 7(1), 83-92.

Runeson, P. (2006). A Survey of Unit Testing Practices. *IEEE Software*, 23(4), 22-29.

Ryan, S. & O'Connor, R. V. (2013). Acquiring and sharing tacit knowledge in software development teams: An empirical study. *Information and Software Technology*, 55(9), 1614-1624.

Scacchi, W. (2005). Socio-Technical Interaction Networks in Free/Open Source Software Development Processes. *Software Process Modeling*, 1-27, Springer.

Schneier, B. (2007). Information Security and Externalities. *European Network and Information Security Agency Quarterly*, 2(4), 3-4.

Shull, F., et al. (2002). What We Have Learned About Fighting Defects. *METRICS '02: Proceedings of the 8th International Symposium on Software Metrics*, 249-258.

Singer, J., et al. (2008). Software Engineering Data Collection for Field Studies. *Guide to Advanced Empirical Software Engineering*, 9-34. Springer.

Sung, P. W.-B. & Paynter, J. (2006). Software Testing Practices in New Zealand. *NACCQ '06: Proceedings of the 19th Annual Conference of the National Advisory Committee on Computing Qualifications*, 273-282.

Takanen, A., et al. (2008). *Fuzzing for Software Security Testing and Quality Assurance*. Artech House.

Thörn, C. & Gustafsson, T. (2008). Uptake of Modeling Practices in SMEs: Initial Results from an Industrial Survey. *MiSE '08: Proceedings of the 2008 International Workshop on Models in Software Engineering*, 21-26.

Tomayko, J. E. & Hazzan, O. (2004). *Human Aspects of Software Development*. Charles River Media.

Tourangeau, R., et al. (2000). *The Psychology of Survey Response*. Cambridge University Press.

Urquhart, C. & Currell, R. (2010). Home uterine monitoring: A case of telemedicine failure? *Health Informatics Journal*, 16(3), 165-175.

van der Merwe, R. (2010). *Investigating direct deliberative governance in online social media*. The Open University.

Vogel, D. A. (2011). *Medical Device Software Verification, Validation and Compliance*. Artech House.

Walker, S. (2008). Digital design in social action settings: A review through a sociotechnical lens. *CIRN '08: Proceedings of the 5th Prato Community Informatics & Development Informatics Conference 2008: ICTs for Social Inclusion: What is the Reality?*

Wohlin, C., et al. (2003). Empirical Research Methods in Software Engineering. *Empirical Methods and Studies in Software Engineering: Experiences from ESERNET*, 7-23. Springer.

Wysopal, C., et al. (2006). *The Art of Software Security Testing: Identifying Software Security Flaws*. Addison-Wesley.