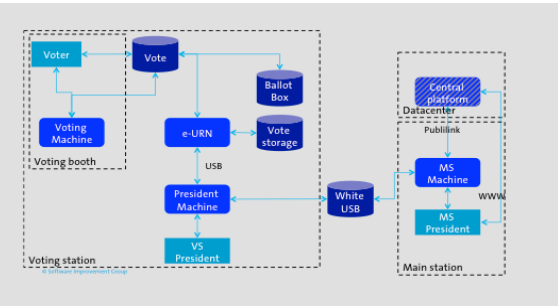




```
int t1s1_process_heartbeat(SSL *s) {
    /* controlled by user */
    unsigned char *p = t1s1->rrec.data[0], *pl;
    unsigned int payload;
    unsigned int padding = 16; /* Use minimum padding */
    ...
    /* The first two bytes of p represent the length of the
     * payload and is put in variable payload
     */
    n2a(p, payload);
    pl = p;
    ...
    unsigned char *buffer, *bp;
    buffer = OPENSSL_malloc(1 + 2 + payload + padding);
    bp = buffer;
    ...
    /* Enter response type, length and copy payload */
    bp++ = T1S1_HB_RESPONSE;
    /* Puts payload length in two bytes of bp. */
    s2n(payload, bp);
    /* Copies payload-length of pl to bp */
    memcpy(bp, pl, payload);
}
```



# Application security of the Belgium electronic voting system

*Rob van der Veer - SIG*



NOT FOR DISTRIBUTION



**OWASP AppSecEU 15**  
Amsterdam, The Netherlands

# Introduction

Rob van der Veer  
Principal consultant  
Software Improvement Group

[r.vanderveer@sig.eu](mailto:r.vanderveer@sig.eu)

[@robvanderveer](#)

+31 6 20437187

[www.sig.eu](http://www.sig.eu)



Software  
Improvement  
Group



NOT FOR DISTRIBUTION

  
**OWASP AppSecEU 15**  
Amsterdam, The Netherlands

# Agenda

- Background
- Publishing source code
- How to analyze source code
- Our approach
- Key take-aways



The screenshot shows a mobile browser interface. At the top, the status bar displays 'vodafone NL 4G 18:06' and '45%' battery. Below the status bar is a navigation bar with a 'Terug' button and the 'datanews' logo. The article title is 'NEDERLANDERS ANALYSEREN BELGISCHE VERKIEZINGSSOFTWARE'. The text of the article discusses SIG's investigation into the source code of Belgian election software used by the Dutch Ministry of the Interior during the 2014 elections. The article is accompanied by a photo of a pen on a document. At the bottom of the article, there are social media sharing icons for Facebook, Twitter, and Email, along with 'T-' and 'T+' buttons for text size adjustment.

13.06.2014 - 16:31

## NEDERLANDERS ANALYSEREN BELGISCHE VERKIEZINGSSOFTWARE

SIG onderzoekt de broncode van de verkiezingssoftware die de FOD Binnenlandse Zaken tijdens de verkiezingen van 25 mei heeft gebruikt bij de stemcomputers en telsystemen.

SIG (Software Improvement Group) pleit al langer voor een doordacht beleid rond technische kwaliteit bij de ontwikkeling van software. Het Nederlandse bedrijf stelt dat kwaliteit het risico op mislukte ict-projecten vermindert. Aandacht voor softwarekwaliteit kan op die manier heel wat geld opleveren. Op 2 juni



# Background

- May 2014: publishing of e-voting source
- European, federal & regional elections
- General e-voting sensitivities
- Sensitivities for Belgium
  - Reliability incidents
  - Usability incidents



NOT FOR DISTRIBUTION

# What happens when you publish source code?



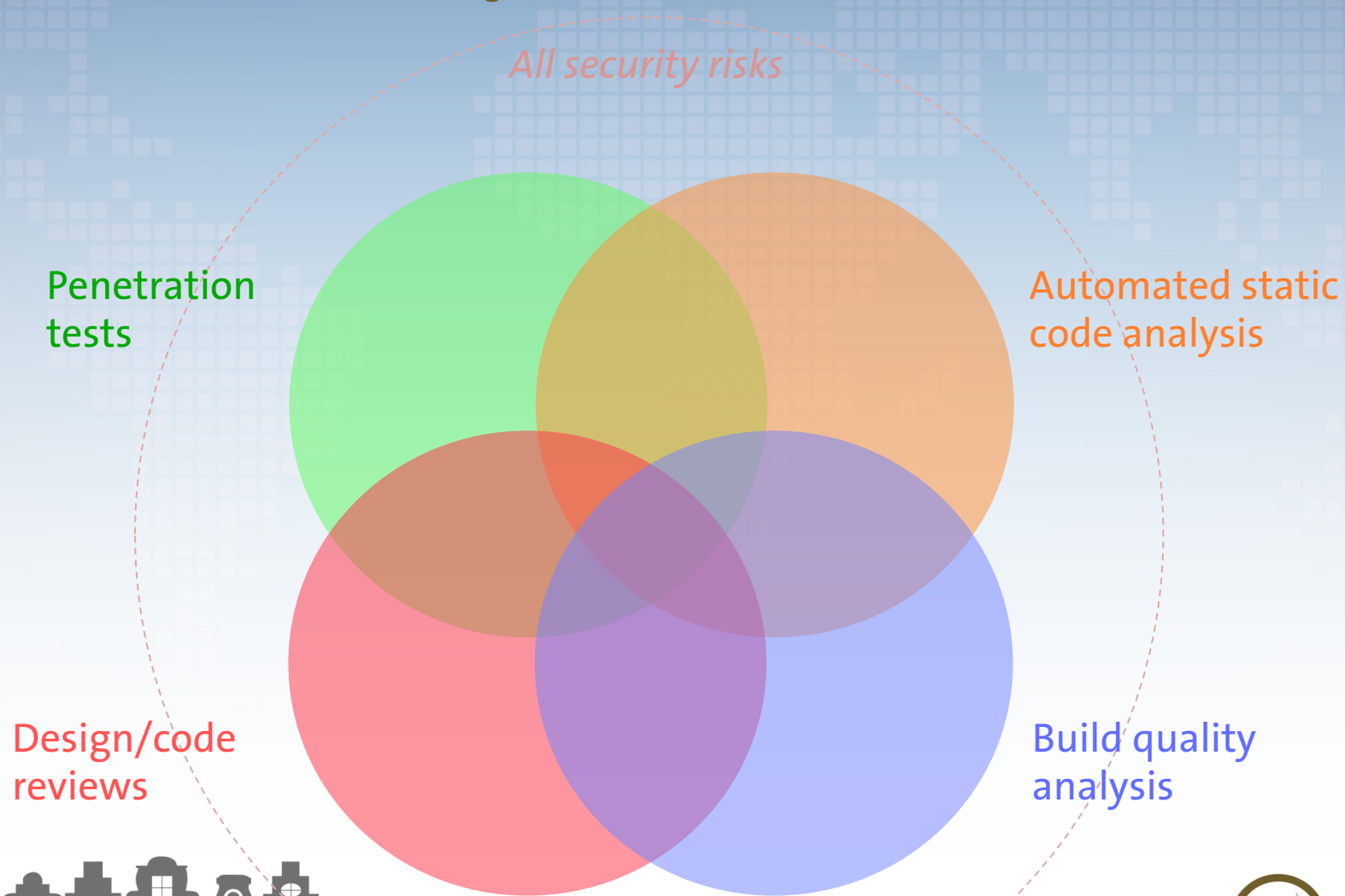
NOT FOR DISTRIBUTION

# What happens when you publish source code?

- It radiates confidence
- People make very little effort to look at it (Heartbleed, anyone?)
- Setting up test environment is hard
- Good code review is hard, especially without interviews
- Code reviewing is sticking your neck out



# How to analyze software security



NOT FOR DISTRIBUTION



**OWASP AppSecEU 15**  
Amsterdam, The Netherlands

# Review system properties to measure ISO25010 security

Final result:  
★★★★★

	Secure data transport	Identification strength	Access management strength	Session management strength	Authorized access	Input and output verification	Secure data storage	Evidence strength	Secure user management	
	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★	★★★★★
Confidentiality & Integrity	X					X	X	X		★★★★★
Non-repudiation & Accountability			X						X	★★★★★
Authenticity				X	X				X	★★★★★





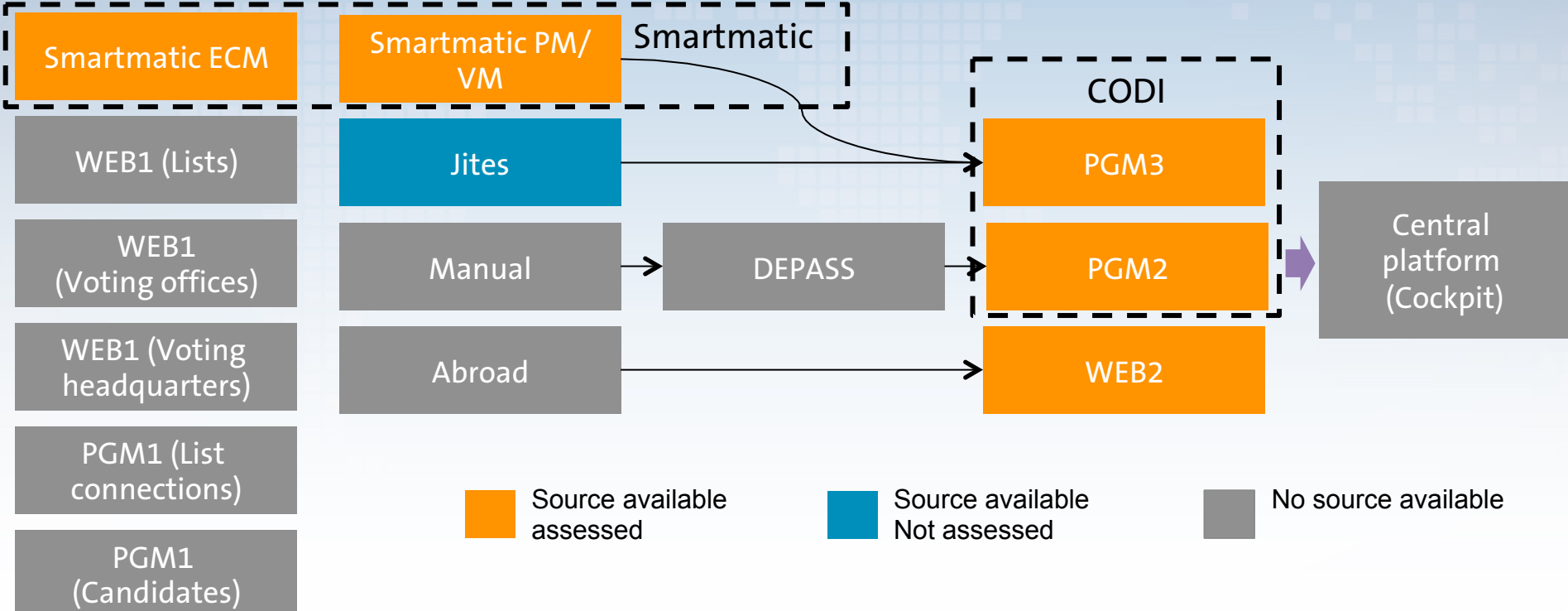
# Measure ISO25010 maintainability

	Code volume	Code quality			Architecture kwaliteit			
	Volume	Duplication	Unit size	Unit complexity	Unit interfacing	Module coupling	Component balance	Component independence
<b>Analysability</b>	X	X	X					X
<b>Modifiability</b>			X		X		X	
<b>Testability</b>	X				X			X
<b>Modularity</b>							X	X
<b>Reusability</b>				X		X		

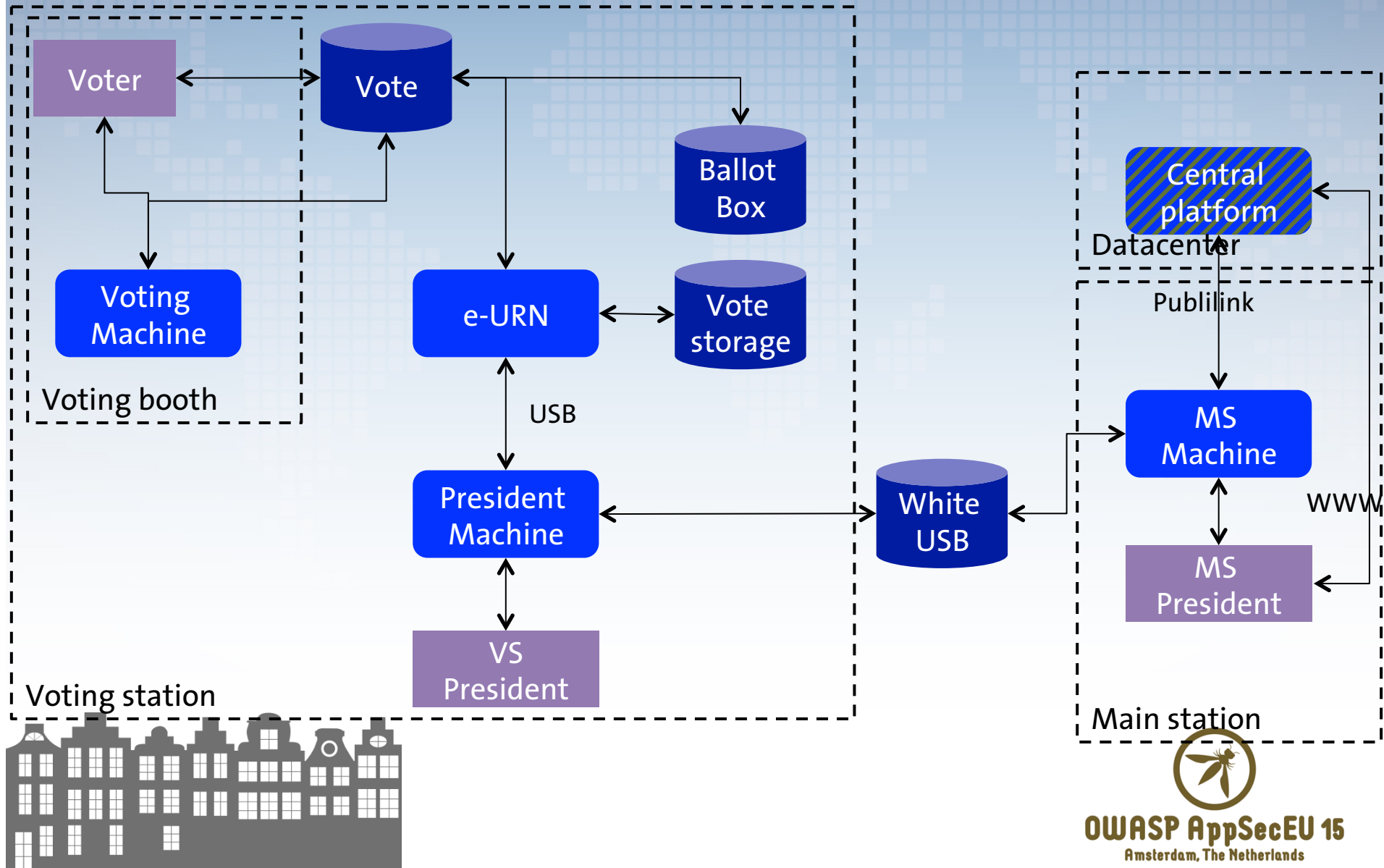


# E-voting software overview

## Voting process



# Threat modeling



# Set SMART non-functional requirements from the start, including who does what

## E-Voting requirements



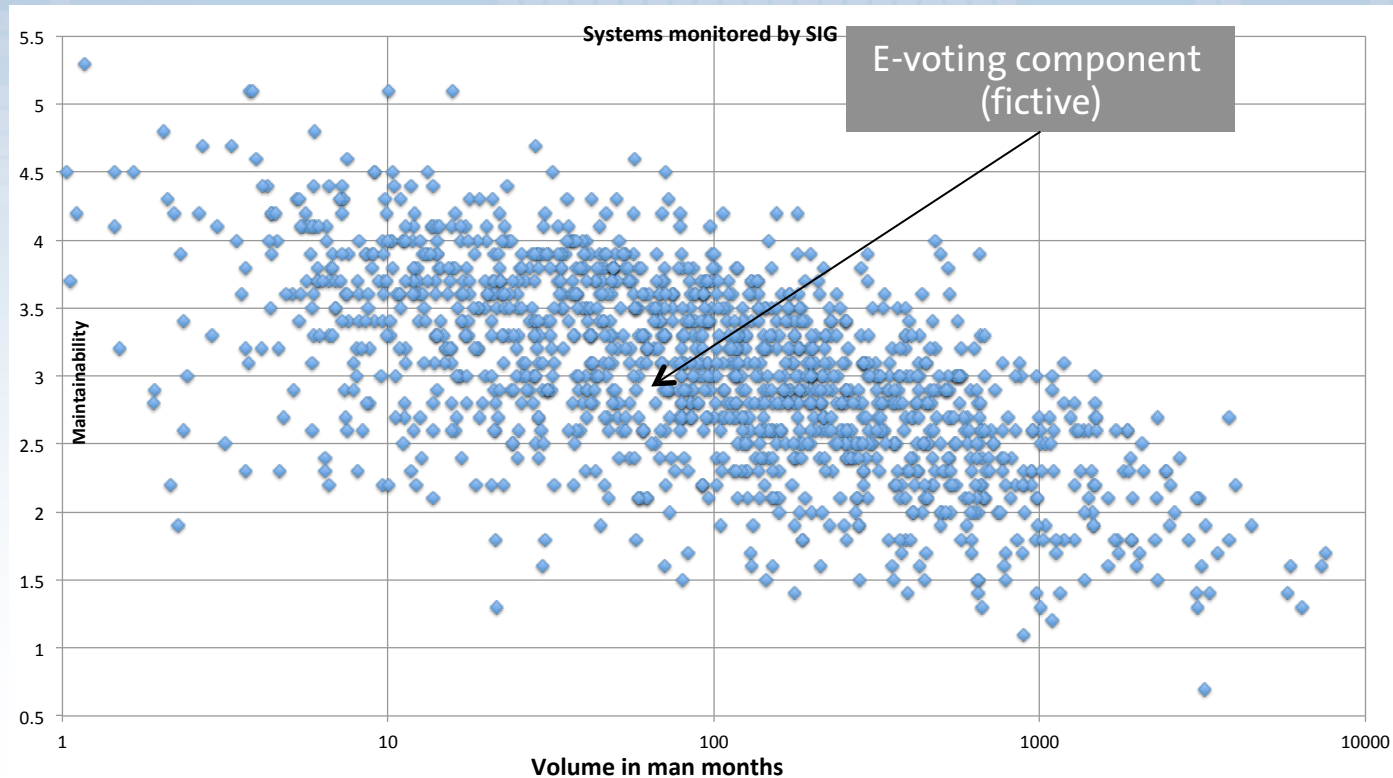
22 functional requirements.

No nonfunctional requirements on building security in

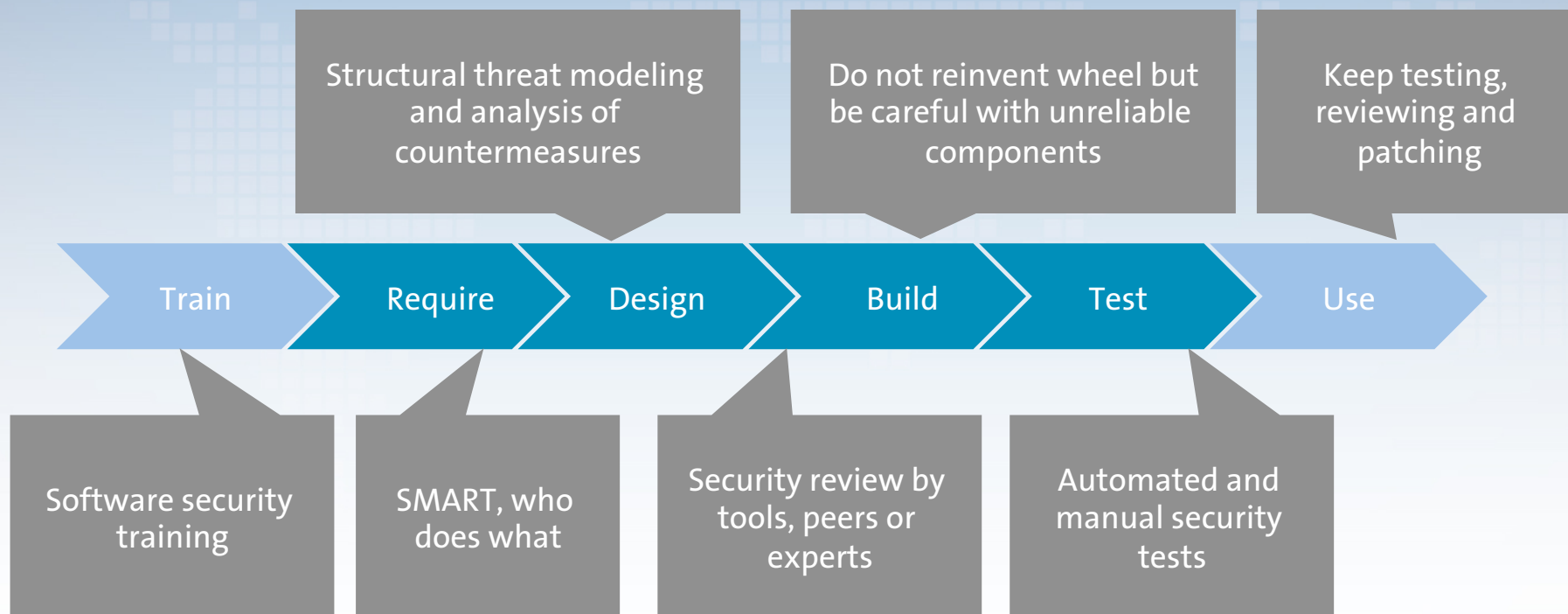
No requirements for other applications in chain



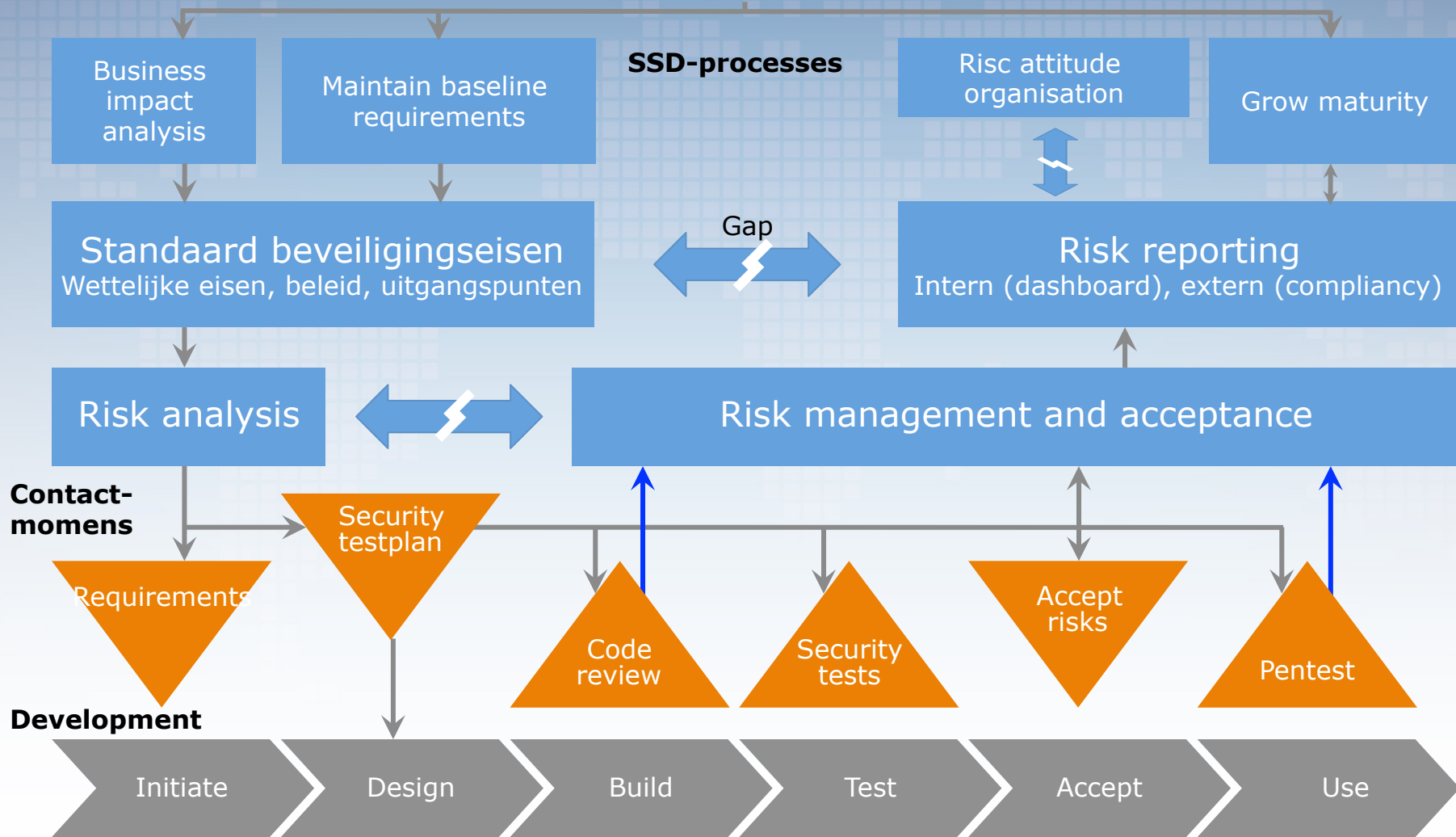
# Also require maintainable software as technical debt fuels security incidents



# Mature the secure SDLC



# Manage your risks



NOT FOR DISTRIBUTION