Notes on "Trustworthy Medical Device Software"

Jonathan Lam

09/09/21

Contents

1 Notes **1**
1.1 Problems 1
1.2 Recommendations for implementing trustworthy medical device software 2
1.3 Policy recommendations for trustworthy medical device software 3
1.4 Vocabulary 3
1.5 Questions 4
by Kevin Fu, UMass Amherst

1 Notes

- What does it mean to be trustworthy? Dependable, customer-responsive, giving assurance, even beyond what the user anticipates? "how well a software system meets *requirements* such that *stakeholders* will *trust* in the operation of the system"; includes but is not limited to "safety, effectiveness, usability, dependability, reliability, security, privacy, availability, and maintainability"
- "A common trait for adverse events in medical device software is that problems are often set in place before any implementation begins"

1.1 Problems

- Problems can occur at many stages in the engineering process, e.g.: requirements specification, design, human factors, implementation, testing, maintenance

- A medical device has to be engineered with consideration to the entire system
- Software requires a different set of tools to assure safety and effectiveness (Pfleeger et al.)
  - Software is discrete (quantization errors)
  - The immaturity of software combined with rapid change
- Antivirus can cause its own problems, namely (incorrectly) blocking critical pieces of software

1.2 Recommendations for implementing trustworthy medical device software

- Medical software is not always following the most up-to-date practices such as requirements specification to verification techniques
  - Other practices include strong typing, static analysis, and a functional specification to be included in the code (Ada seems to be a good choice)
- Proprietary software can cause problems with software security: it makes it harder for researchers to access it; the NRC report recommends "direct evidence to support claims about software dependability"
- Software needs to be designed w.r.t. the entire system and anything that it might interface with, which may include software/hardware from other companies (integration tests?)
- Mitigate risks due to human factors: bad UI/UX (labels/instructions, user input validation), targeting the incorrect audience (e.g., doctors vs. nurses)
- Mitigate low-probability, high-consequence risks: don't be so sure that low-probablity risks won't happen
  - Difficulty of reproducing a bug may cause dismissal of the bug, rather than finding its root cause (e.g., via RCA)
  - Healthcare professionals should know about all risks, no matter how small
  - Unspecified constraints can be exploited

1.3 Policy recommendations for trustworthy medical device software

- Specify outcome measures, not technology
- Collect better statistics on the role of software in medical devices: current data-collection tools are severely underutilized, and databases are severely underreported
- Enable open research in software-based medical devices: reduce proprietaryness of the medical software industry
- Clearly specify roles and responsibillity: fine-grained roles for specific users
  - This may be harder for monolithic systems, especially ones that go out-of-date. Commercial Off-The-Shelf (COTS) systems may have a shorter lifetime than medical equipment, which is expensive and may last several decades. Thus old software may be required to operate old hardware. Hospitals may have problems with old operating systems and incompatible software.
- Clarify the meaning of substantial equivalence for software
  - Example: EMR a.o.t. paper records
- Increase FDA access to outside experts in software engineering

1.4 Vocabulary

- Trustworthy (see first part)
- Threat: a set of circumstances that has the potential to cause loss or harm; we say threats exploit a vulnerability
- Vulnerability: a weak spot that may cause a threat to be realized
- COTS: Commercial Off-The-Shelf (usually referring to software)
- Substantial equivalence: a method of quickly approving new medical devices by likening its use to a previous device (a "predicate")

1.5 Questions

- Is the article talking about unit and integration tests? They don't use the terms there.
- Relationship between OSS/proprietary software and security?
- RCA? What bugs can/can't static analysis help prevent?