Vom Papier zur normkonformen Mobile Medical App – Ein Erfahrungsbericht
we engineer medical software
welcome
MedConf2019

Bernd Seidenspinner
Lead Software Architect

Miriam Schulze
Director Medical Engineering
we are certified by TÜV Hessen according to ISO 9001 and ISO 13485 and engineer software in accordance with IEC 62304
BAYOOMED at a glance

the leading software development company for regulated development of medical apps and medical (standalone-) software

most experienced
250 person years experience in medical software development
> 800 medical customers

medical software
mobile medical apps for iOS (iPhone & iPad) and Android.
device classes I, IIa, IIb and III
software safety classes B & C
cloud & web applications

Darmstadt & Munich
widespread expertise
great passion

100+ employees

ISO 13485
the beginning
Even the smallest contacts in the μ-range can cause dysfunctions (usually pain dysfunction syndrome) in patients and even significantly hinder swallowing. The representation of the occlusal contact conditions therefore places a very high requirement on the test equipment.

goal
digitalization of product and process

system: handheld + iPad app
combination analog and digital
more than just a digital pedant to the analog product
→ added value for the dentist
let’s do it agile, please
scrum: process & advantages

scrum imposes a few constraints on the development process

- empirical process control: transparency, inspection, adaption
- fixed time boxes
- results every 3 weeks
implementation

scrum: process & advantages

sprint backlog

sprint planning

daily scrum

do

sprint review

improve

sprint retrospective

product backlog

BAYOOMED development team

Customer stakeholder

BAYOOMED product owner

BAYOOMED scrum master

finished product
Agile?!
What about documentation

1. "Definition of Done" includes documentation, verification and tracing activities
2. Creation and maintenance of software development plan according IEC 62304
3. Documentation plan contains relevant documents, templates and storage locations

Acceptance Criteria

<table>
<thead>
<tr>
<th>B</th>
<th>I</th>
<th>U</th>
<th>A</th>
<th>☑</th>
<th>☑</th>
<th>☑</th>
<th>☑</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOD:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: Software Architecture and Design was created in Enterprise Architect</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: User Story was developed according the given software design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: Software Architecture and Design documentation are in sync with the implementation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: User Story was developed according the given UI design and styleguide</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: All requirements and use cases linked to the User Story were covered</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: Unit test specification (inline in code) was defined prior implementation of unit test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: All unit test were implemented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: In case functionality was not unit tested, the reason for that decision was documented</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: All User Story specific acceptance criteria were met</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: All unit test results have the status &quot;passed&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: Feature and test code, specification and documentation is committed to GIT and linked to the correct TFS task</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: Error Logging is implemented where applicable</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USER STORY: User Story Status is updated accordingly</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PROJECT: Identified software risks were documented and a mitigation was established, when possible</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMPLEMENTATION: BAYOONET code and GIT guidelines were met</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
specify, design, implement, verify
implementation tool chain

specify

Enterprise Architect for requirements management

Word and Excel for test plans, memo, strategy paper, etc. using existing templates

design

Enterprise Architect for sw architecture and design

tracing is done in Enterprise Architect as well

implement

GIT for revision and code control

DevOps for sprint organization

feature and test development in XCode

verify

GIT in combination with DevOps and separate build server for automated tests
implementation
CI/CD pipeline iOS app

MS DevOps

development
macOS & XCode
git
iPad / iOS
Safari browser

build & (unit)test
macOS VM
XCode / iOS emulator
git
build agent

release
Windows VM
release agent

QA Testflight
internal QA / validation

staging Testflight
customer limited number of beta Testers

AppStore
released version

BAYOOMED
MEDICAL SOFTWARE
iOS - lessons learned
Apples update strategy causes challenges:
- AppStore policy change → causes need for updating XCode
- causes need for updating macOS

integration of DevOps & iOS & macOS was less painful than expected

AppStore distribution for releases remains manual process
there is only very limited use of testing the app with beta versions of iOS

→ better to invest efforts into higher test automation
development process
lessons learned

- no iPad mini support: no device type detection possible
  → undocumented API calls

- missing high level API for network communication
  → custom C code and SOUPs
versioning of subsystems imposes changes to repository structure (if gitflow is used)

→ independent versioning of app and cloud
→ independent (linked) git repositories
just upload it to the app store
“Your app includes content or features that users aren’t able to use in this version. Apps that are for demonstration, trial, or up-sell purposes are not appropriate for the App Store.”
→ “Demo”-mode → unregistered

“We have started the review of your app, but we are not able to continue because we need access to a video that demonstrates your app: in use on a physical iOS device, which shows the pairing and interaction with the designated hardware. Please ensure the video you provide shows a physical iOS device (not a simulator)”
→ no need for sending a handheld → video

“We noticed that your app has iTunes File Sharing enabled.
- Which features of your app require iTunes File Sharing?
- What type of files would users normally be transferring with iTunes?
- Where specifically in the app would users access these shared files?”
→ detailed description of airdrop usage

“Which specific features of the app utilize audio functionality while the app is running in the background? Where can these features be located within the binary?”
→ deactivation
yet, another update
updates

how to handle them

part of post-market
surveillance activities

new iPads
check iPad versions within the application
⇒ lock app if required
no restrictions in app store possible

new handheld firmware
compiled into app
creation of new app version for firmware updates
⇒ app store guidelines compatibility check

new iOS
cloud db containing compatibility information
unblocking after verification (automated and manual tests)
discussion
thank you
versioning of subsystems imposes changes to repository structure (if gitflow is used)

- independent versioning of app and cloud
- independent (linked) git repositories
development process
lessons learned

- no iPad mini support: no device type detection possible
  → undocumented API calls

- missing high level API for network communication
  → custom C code and SOUPs
implementation

CI/CD lessons learned

Integration of DevOps & iOS & macOS was less painful than expected.
AppStore distribution for releases remains manual process.

Apple's update strategy causes challenges:
- AppStore policy change
  - causes need for updating Xcode
  - causes need for updating macOS