Bridging the Gap Between Development and Regulatory Teams

Milton Yarberry
Director of Medical Programs, ICS

MEDICAL DEVICE QUALITY IS ALL WE DO, AND WE'RE ALWAYS AHEAD OF THE GAME.

1.5M years industry experience

522k podcast listeners

182k+ look to us for the latest in quality

#1 blog and podcast in the industry

750+ **I-III/SaMD/IVD** medical device companies worldwide in all device classes and types

1700+ 510(k) clearances & CE marked devices

2000+ ISO 13485 certification

1000s customer approvals and audits passed

TRUSTED BY LEADING MEDICAL DEVICE COMPANIES GLOBALLY

G2 CROWD LEADER MEDICAL QMS SOFTWARE ★★★★☆

G2 CROWD LEADER QMS SOFTWARE ★★★★☆

"Best eQMS I have ever used..."

This is the easiest eQMS I have used in the 20 years I have been in the Medical Device Industry. ***It is simple, intuitive and easy to use...*** We are successfully implementing a Quality Culture.

- Director of Regulatory Affairs & Quality Assurance

"Modern QMS Software and Outstanding Customer Service."

★★★★★

"Demystifying QMS and Regulatory Requirements"

★★★★★

"Makes your QMS Simple and Effective"

★★★★★

greenlight guru

About ICS and Boston UX

Creating Transformative Products That Advance Patient Care

Established in 1987, ICS delivers innovative medtech solutions with a full suite of services to accelerate development, testing and certification of successful next-gen products.

ICS and Boston UX are headquartered in Waltham, Mass. with offices in California, Canada and Europe.

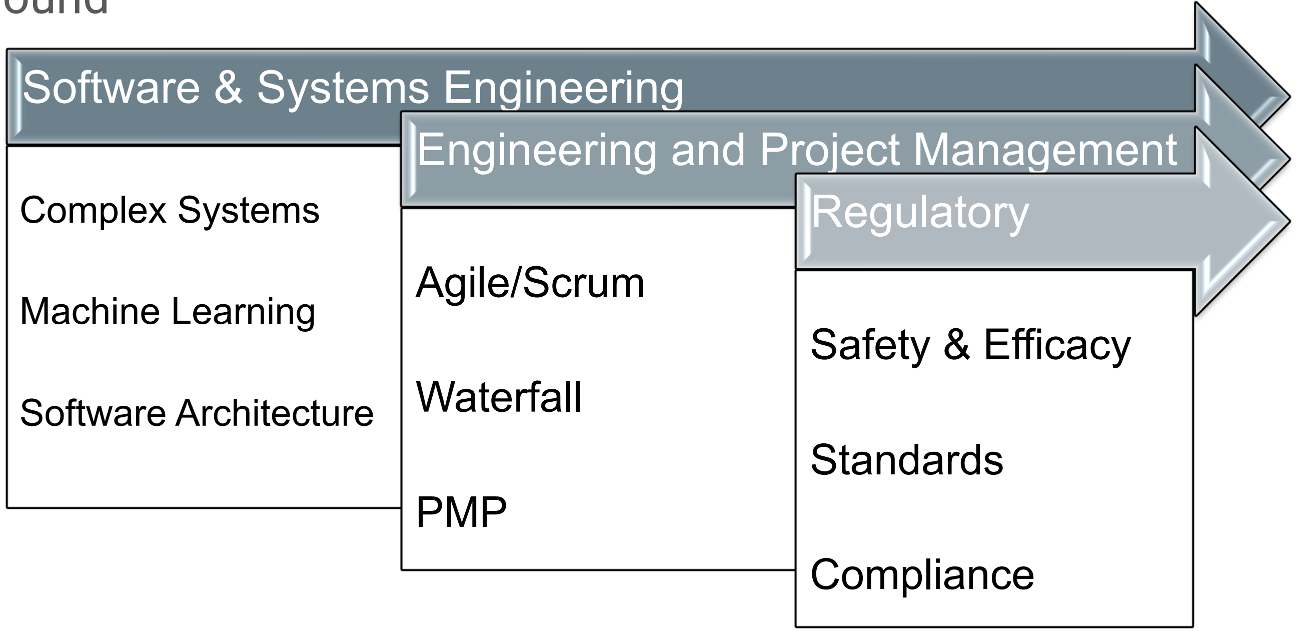
ICS' design studio specializes in intuitive touchscreen and multimodal interfaces for high-impact embedded and connected devices.

Delivering a Full Suite of Medtech Services

- Human Factors Engineering
- IEC 62366-UX/UI Design
- Custom Frontend and Backend Software Development
- Development with IEC 62304-Compliant Platform
- Low-code Tools that Convert UX Prototype to Product
- Medical Device Cybersecurity
- AWS and Azure Cloud Services and Analytics
- ISO 14971-Compliant Hazard Analysis
- Software Verification Testing
- Complimentary Software Technology Assessment

Development → Regulatory

My background



Development → Regulatory

Agenda

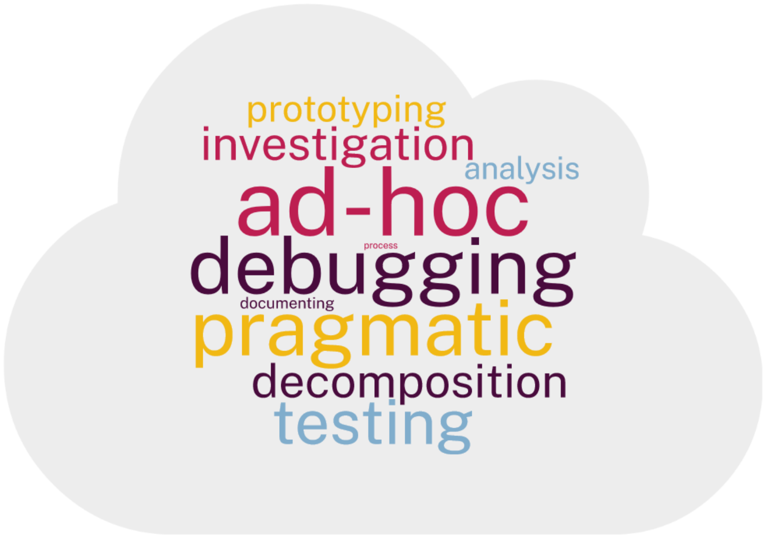
1. Defining the problem – the Gap
2. Complicating Factors
3. Bridging – the Gap
4. Nuances
5. Summary

Defining the Problem

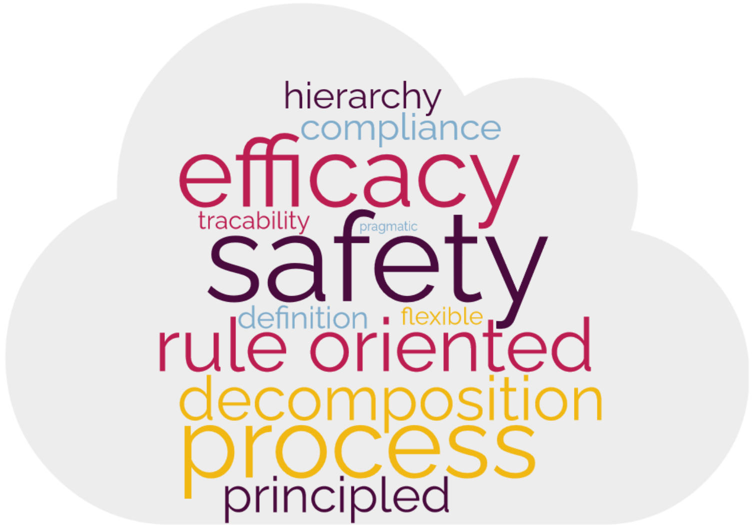
Defining the Problem

Native Characteristics Cloud

Development



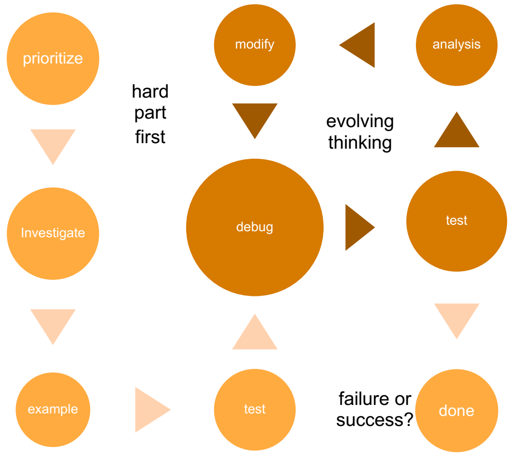
Regulatory



Defining the Problem

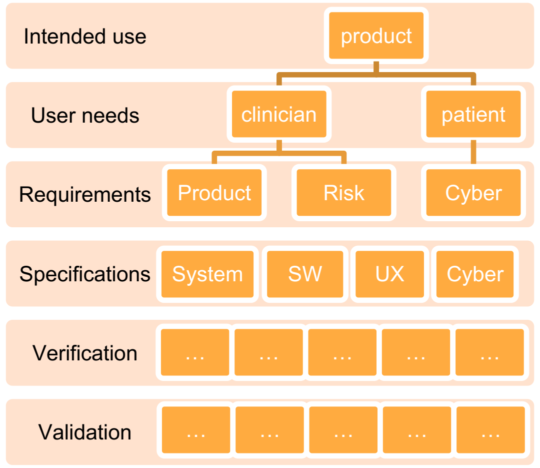
Native Workflow

Development



- Do a bit of everything
- Iterate towards a solution
- Discovery
- Result driven, dynamic process

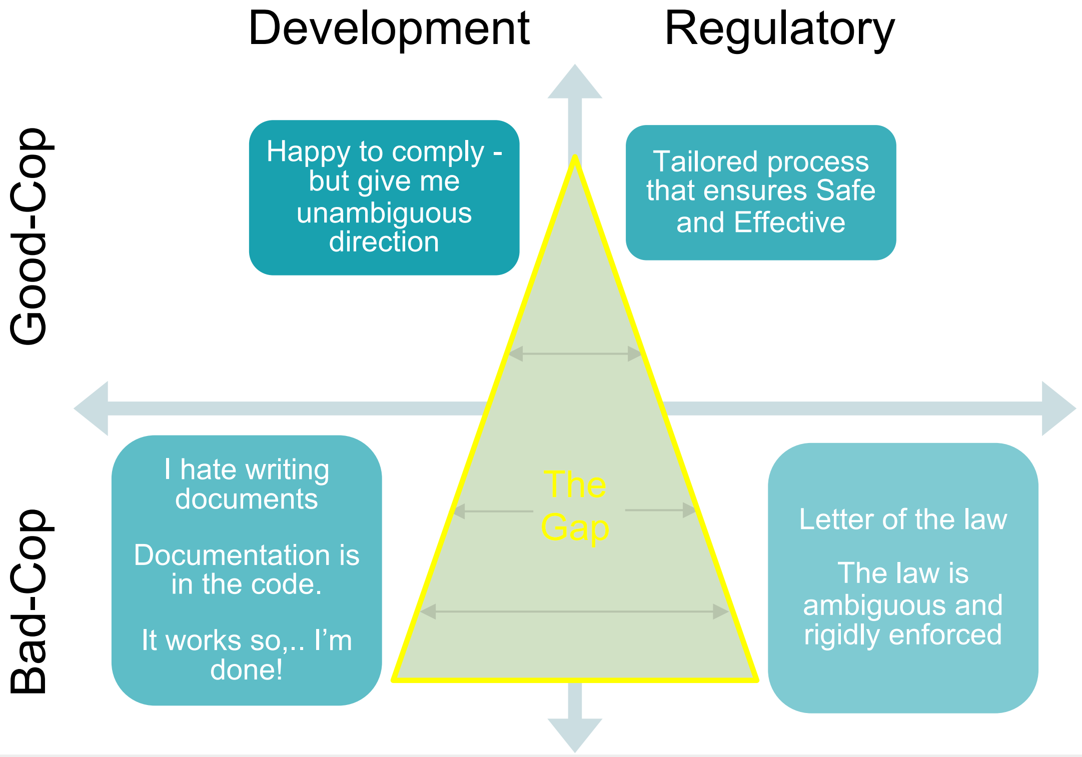
Regulatory



- Start at the top and trace down
- Hierarchical
- Phases
- Defined, static process

Defining the Problem

The GAP


Development
Regulatory
Good-Cop
Bad-Cop
Happy to comply - but give me unambiguous direction
Tailored process that ensures Safe and Effective
I hate writing documents
Documentation is in the code.
It works so,.. I'm done!
The Gap
Letter of the law
The law is ambiguous and rigidly enforced

Defining the Problem
Logical Gaps

Common events that exacerbate gaps:

Requirements

- Waiting for detailed Product Requirement decisions to be made
- Conflicting input – stems from no single source of truth that's widely used
- No timeline for answers – no commitment to conclusion stalls progress and isn't visible
- Lack of certainty about what level to document requirements – what's essential for your Intended Use

Discovery

- Lack of deep understanding of corner cases – error recovery is always a deep topic that is often misunderstood/underestimated
- Deferring discovery – pushing prototyping efforts into middle schedule

Single source of truth

- Not knowing what's approved vs. under discussion
    i. Aligning MRD vs. everything else
    ii. Document Control System
    iii. Distributing approved documentation

Complicating Factors

Complicating Factors

Lagging process

Process-lag

- Often/usually/always? Engineering is active before the QMS is approved
- Starting development without a QMS in place creates ambiguity
  - Creates a need to 'catch-up'
  - Process-debt → confusion

Example: when should design documentation begin?

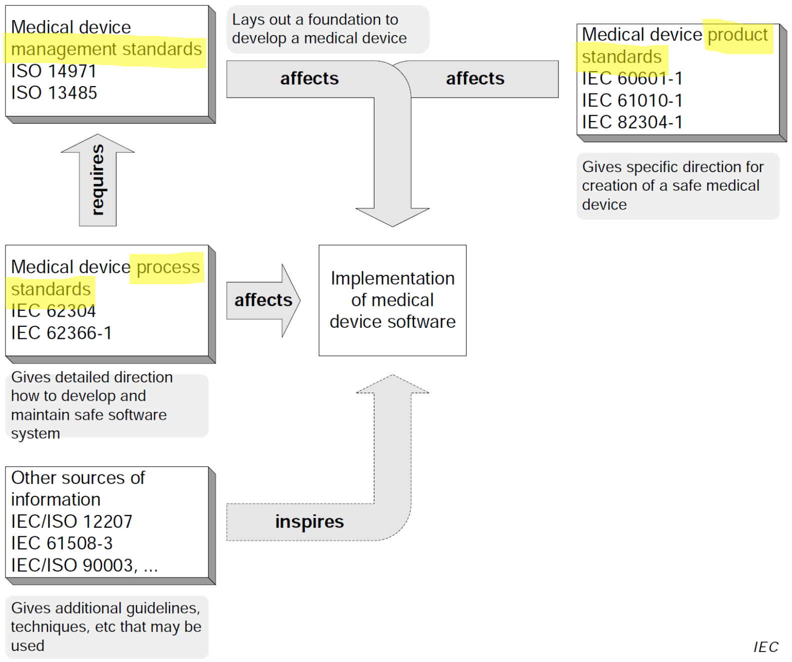
2 rules of thumb:

1) When you're developing 'product' (not prototyping)
2) After the product requirements are approved

But,

1) Is there a precise point when you stop prototyping?
2) Product requirements are often evolved

Complicating Factors
Ambiguous process



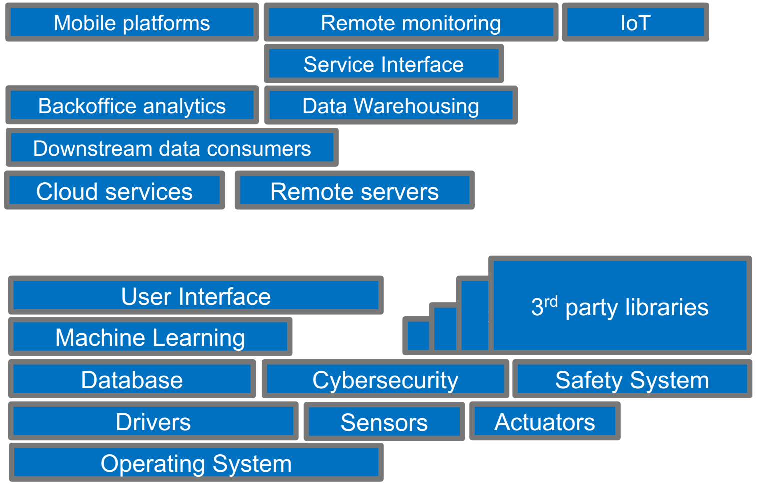
Medical device management standards
ISO 14971
ISO 13485
Lays out a foundation to develop a medical device
affects
affects
Medical device product standards
IEC 60601-1
IEC 61010-1
IEC 82304-1
Gives specific direction for creation of a safe medical device
requires
Medical device process standards
IEC 62304
IEC 62366-1
affects
Implementation of medical device software
Gives detailed direction how to develop and maintain safe software system
Other sources of information
IEC/ISO 12207
IEC 61508-3
IEC/ISO 90003, ...
inspires
Gives additional guidelines, techniques, etc that may be used
IEC

Process-autonomy

- FDA regulations contain no specifics.
  → WHY?
- FDA wrote the regulations to promote **flexibility** for manufacturers
- Manufacturer's obligation to understand their own product, environment, application and risks
- Autonomy = process tailoring = ambiguity

Complicating Factors
Complexity and late discovery

Software Stacking



- Unrestrained Complexity / Staggering amount of content
- Late discovery of technology issues can impact non-adjacent layers – hugely disruptive
  i.e. technology replacement
- Late discovery is inevitable, but the quantity and impact can be minimized
- This effect increases in the future
  Number of layers
  Depth of complexity
  = **Geometric complication**

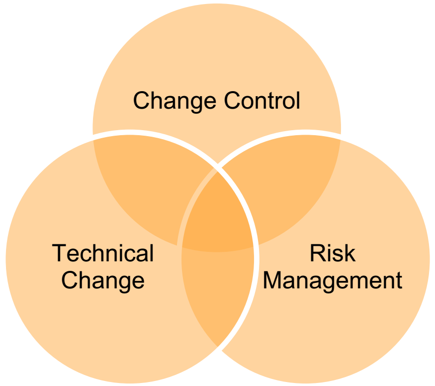
Complicating Factors
Managing change

The change process is non-trivial

Pre-V&V (Verification & Validation)

- Pre-V&V is less formal, but
- The change process is variable and very messy
- Modification, approval, tracing of Design Outputs

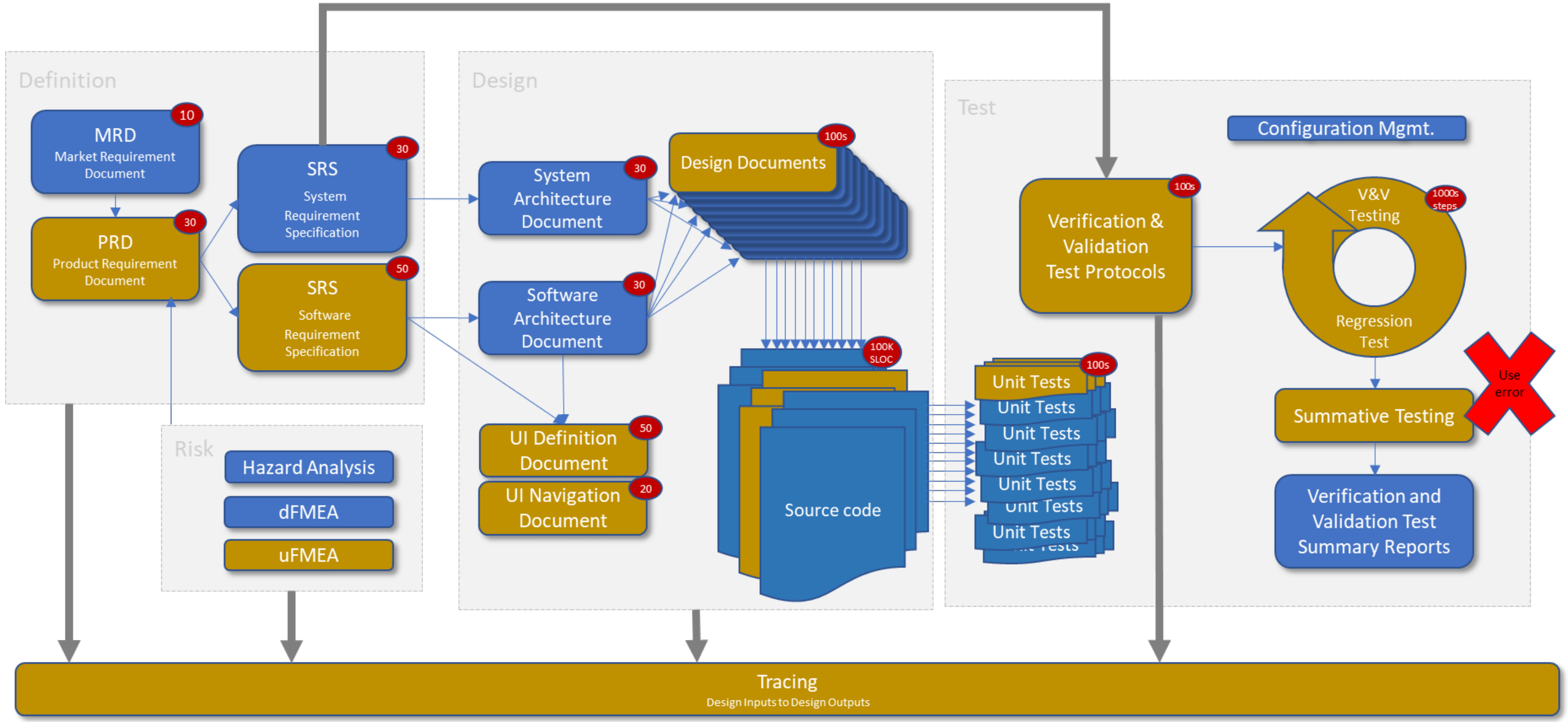
Post-V&V: Overlap of change considerations



►►► ***Changes ripple through design collateral***

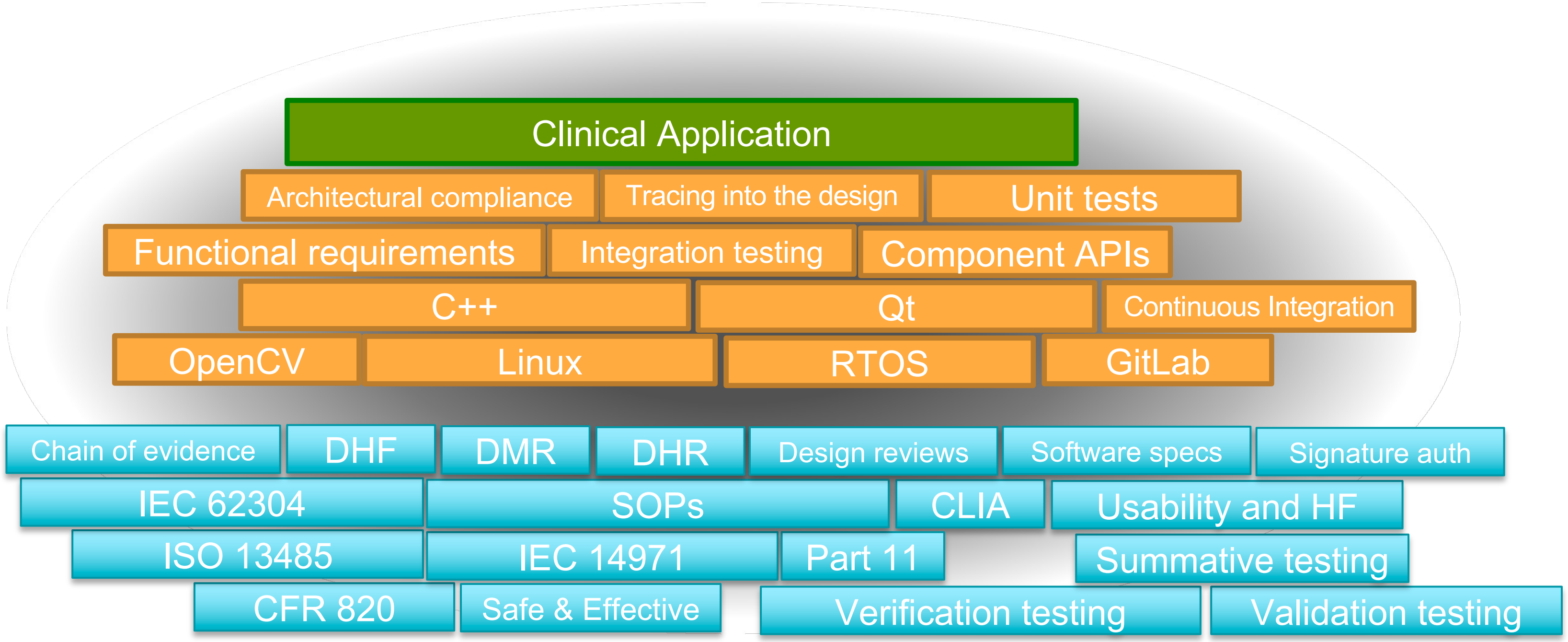
Complicating Factors

Managing change – A Use Error Example


Definition
MRD
Market Requirement Document
10
PRD
Product Requirement Document
30
SRS
System Requirement Specification
30
SRS
Software Requirement Specification
50
Risk
Hazard Analysis
dFMEA
uFMEA
Design
System Architecture Document
30
Software Architecture Document
30
Design Documents
100s
UI Definition Document
50
UI Navigation Document
20
Source code
100K SLOC
Test
Configuration Mgmt.
Verification & Validation Test Protocols
100s
V&V Testing
1000s steps
Regression Test
Unit Tests
100s
Summative Testing
Use error
Verification and Validation Test Summary Reports
Tracing
Design Inputs to Design Outputs

Complicating Factors

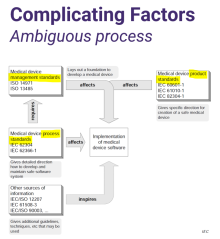
Cognitive Saturation – software engineer



Layered knowledge and constraints saturates an individual's cognitive capacity.

Bridging

Bridging the Gap
Process lag & Ambiguous Process


Complicating Factors
Ambiguous process
Medical device management standards
ISO 14971
ISO 13485
Lays out a foundation to develop a medical device
affects
affects
Medical device product standards
IEC 60601-1
IEC 61010-1
IEC 82304-1
Gives specific direction for creation of a safe medical device
requires
Medical device process standards
IEC 62304
IEC 62366-1
affects
Implementation of medical device software
Gives detailed direction how to develop and maintain safe software system
Other sources of information
IEC/ISO 12207
IEC 61508-3
IEC/ISO 90003, ...
inspires
Gives additional guidelines, techniques, etc that may be used
IEC

Don’t create a gap

- Define (document, approve, distribute) design expectations when developments starts
  - Interim Development Plan to avoid a gap
  - For example, start with a ‘prototyping process’
  - If no prototyping requirements, then define the boundary to prototyping
- Size the plan to the risk of the development activity
- **Grow** the plan with new activities
- Make the process explicit, simple *(work instructions)*

Implement a QMS progressively

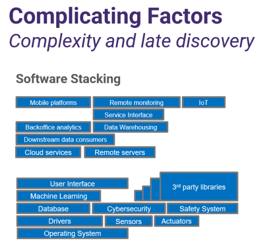
- Use a risk based approach to prioritize QMS procedures
- Start with: Quality Manual, Design Control, Document and Records, Risk Management
- Add as they become relevant

KISS - Keep it simple/stupid

Use tools early – don’t invent and migrate

- Starting in a spreadsheet or simple document and migrating later tends to be vastly inefficient

Bridging the Gap
Complexity and late discovery


Complicating Factors
Complexity and late discovery

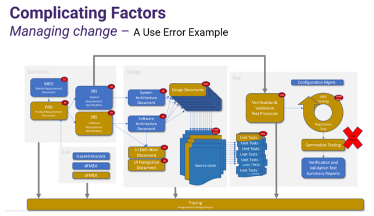
Software Stacking

Assume an environment of change
- Don't treat change as the exception
- Build technology changes into your process
  - i.e. change image library

Write plans to minimize impact of change
- Aggregate design reviews for a sub-component

Tight cross-functional development
- Changes to HW can impact SW and vice-versa
- Incompatibilities drive late discovery
- Design reviews should have exhaustive input from adjacent development groups
  - Mechanical, Usability, Systems, Electrical, Software

Bridging the Gap
Managing change



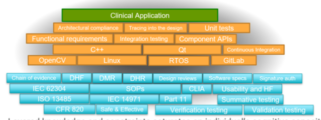
The obvious: up-front diligence and discovery is better than late-stage testing (or discovery in the field)

Manage Change by minimizing Change

- Analyze the risky or complex components
  - Prototype testing (if you haven't tested it, assume the packaging doesn't match the contents)
  - Detailed review (cursory buy-in is the best place for late stage disaster to hide)

Explicit, simple, processes reduce the effort for changes

Complicating Factors
Cognitive Saturation – software engineer



Layered knowledge and constraints saturates an individual's cognitive capacity.

Bridging the Gap
Cognitive saturation

Make fewest demands feasible

- Have regulatory drive development formalization – don't rely too heavy on engineers

Simplest possible process

- Think in terms of bare minimum process, but tailor them to the product need
- Avoid concessions to 'no value, but satisfies compliance'

Consider Work Instructions for the most complex or common tasks

Nuances

Nuances: Single source of truth

The challenge: working-on and accessing approved project documents

File shares

- No Part 11 compliance
- No workflow management
- Weak versioning
- Weak version control
- Weak audit support
- No tracing

Document Control Systems

- Database like – not folder centric
- Access to approved documents license-limited
- Weak work-in-progress management
- Encumbered interface
  - Results in lagging updates
  - Offline caching
  - Not the centralized source of truth

Leads to

- Uncertainty
- Mistakes
- Reworks
- And poor traceability

Solutions

- Design a document process optimized for
  - Broad and easy access to approved documents
  - Easy review/approval mechanism
  - **Account for work-in-progress**
  - Audit trail
- Use customized QMS tools

Nuance: Development Prerequisites

When does development start? When to turn on Design Controls?

Why do we need Design Controls?

- Meet the needs of end user and patients
- Ensures Intended Use is achieved
- Prevent unintended behavior in the delivered product
- Ensures risks are managed

Very little needed to start Design Controls:

- Development Plan (responsibilities, activities: definition/design/V&V, design outputs, etc.)
- Product Requirements (design inputs)
- QMS Procedures / SOPs?

But, when should we start Design Controls?

- When product development has started

Are we still prototyping?

- Is any part of the prototype going to be used in the final product?

  Yes → developing
  No → prototyping

Risks without Design Controls

- Developed the wrong functionality
- Leaving unintended features in the product
- No design review of prototypes

Nuance: Leveraging prototyping
Acceleration Opportunity

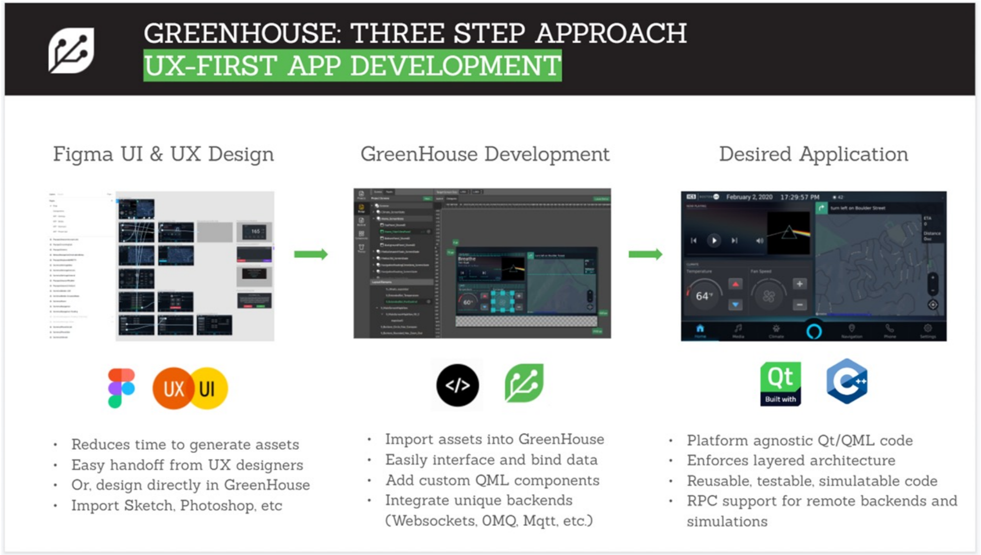
Prototyping can de-risk late-stage development

Use rapid-prototyping UI tools to:

- Define the User Interface
- Circulate & Collaborate
- Explore corner cases
- Approve a versioned instance
- Export to a prototype on the target processing and display hardware

High-fidelity prototyping tools can:

- Enable pixel perfect exports
- Maintain fidelity from wireframes to backend functions


GREENHOUSE: THREE STEP APPROACH
UX-FIRST APP DEVELOPMENT

Figma UI & UX Design
- Reduces time to generate assets
- Easy handoff from UX designers
- Or, design directly in GreenHouse
- Import Sketch, Photoshop, etc

GreenHouse Development
- Import assets into GreenHouse
- Easily interface and bind data
- Add custom QML components
- Integrate unique backends (Websockets, 0MQ, Mqtt, etc.)

Desired Application
- Platform agnostic Qt/QML code
- Enforces layered architecture
- Reusable, testable, simulatable code
- RPC support for remote backends and simulations

In a nutshell

In summary – *Bridging Development and Regulatory*

Different native tendencies

Incompatible workflows

Diverse set of personalities

+ Huge range of complicating factors

GAP: Functional, Cultural (and notorious)

In summary – *Bridging Development and Regulatory*

1) Don’t start with a Gap: when developers start, create a micro-process to cement expectations (benefits: early process patterning, reduced ambiguity, no conversion waste, less design refactoring)
2) KISS – keep process obligations simple, obvious
3) Regulatory-led process steps (Support developers Use Work Instructions where needed)
4) Leverage prototypes (Use development tools that leverage prototypes into production code)
5) Turn on Design Controls only when developing production code
6) Carefully optimize the document management process (easy access/review/approval and WIP)
7) Use customized QMS tools early – avoid conversions

Questions?