DISCUSSION DRAFT v1.8 for FDASIA Subgroup on Framework for Risk and Innovation DIMENSIONS of ASSESSING RISK of PATIENT HARM

|  |  |  |  |
| --- | --- | --- | --- |
|  | Lower risk | Medium Risk | Higher Risk |
| Purpose of software product | Information-only; purpose is transparent and clear | Makes recommendations to user | Automated decision making (e.g., intelligent IV pump, AED) |
| Intended user(s) | Targeted user(s) are knowledgeable and can safely use product | Makes recommendations to patients | Provides diagnosis or treatment advice directly to patient |
| Severity of risk | Very low probability of harm | Potential for non-life threatening adverse event | Life-threatening potential |
| Number of people exposed | Software relevant to few people (<100) | Moderate-sized population exposed (100-1000) | Software relevant to large populations (>1000) |
| Likelihood of risky situation arising | Rare  (<1 per 100,000 patient-years) | Unpredictable, but risky situation arises > 1:100K pt-yrs and < once a year | Common (arises once per patient-year) |
| Transparency of software operations and data and included content providers | Software output is easy to understand and its “calculation” (data and algorithm) transparent | Software operates transparently and output is understandable by software expert | “Black box” |
| Ability to mitigate harmful condition | Human intermediary knowledgeable and empowered to intervene to prevent harm | Human intermediary may be (but not routinely) involved | Closed loop (no human intervention) |
| Complexity of software and its maintenance | Application of mature, widely adopted technologies with information output that is easy to understand by the user | Medium complexity. Testing procedures exist that reliably assess patient-safety risk profile of product. | Complexity of data collection and “transformation” involved in producing output is significant. Difficult to test reliably for all safety risks |
| Complexity of implementation and upgrades | The “build” and configuration of the software is straight-forward and does not materially affect the integrity of the output | The “build” and configuration of the software is moderately complex, but “guard rails” significantly limit types of changes that might induce life-threatening risk. | The “build” and configuration of the software is complex and can introduce substantial changes that can induce serious risk. Limited or no “guard rails.” |
| Complexity of training and use | The software system output is clear and easy to interpret. Minimal training needed. | Moderate complexity. Less than 1 hr of training required. | The complexity of the user interface and density of data presented can cause important errors or oversights that can lead to serious risk. Formal training necessary. |
| Use as part of more comprehensive software/hardware system | Used as a standalone product, or output is unambiguously used as part of larger integrated system. Certified to specific hardware. Redundancy reduces single points of failure | Software interacts with 1-3 other systems with mature, well described interfaces | Almost always used as part of a larger software system AND output is subject to interpretation or can be configured in multiple ways whose mis-interpretation may induce harm. [e.g., DDI thresholds]. |
| Network connectivity | Wired or tightly controlled wireless spectrum | Unregulated spectrum, but low risk of interference | Wireless using unregulated spectrum |

DRAFT Definitions

**Purpose of software** – the intended purpose (and users) for the software, as declared by the developer

* Developer provides transparent purpose and “scope of intended use”
  + For limited scope systems (e.g., radiation planning for use by radiation oncologist), would reduce the burden of complying with any regulation
  + For limited applications (e.g., information only for patients/consumers), it may effectively waive consideration for regulation
  + Regulatory language could control “off-label” use
* By transparently declaring intended purpose, FTC may be able to hold developer accountable to stated purposes

**Intended users** – the intended users of the software, as declared by the developer

* The usability, ability to understand and act on the software output by the intended user is considered in the risk of the software’s use in contributing to patient harm
* The risk assessment would be applied to each class of intended user

**Severity of risk** – the seriousness of patient harm that might arise from appropriate use of the software

* Patient harm is an adverse event resulting from use of the software, which could vary in severity from a life-threatening event to a non-life-threatening adverse event
* Risk could arise from anticipated, appropriate use or from foreseeable inappropriate use

**Number of people exposed –** number of patients whose care is likely to be exposed to the potential risk, measured in patient-years

* Combination of the number of patients with the relevant condition (e.g., diabetes, hospitalized, specific cancer therapy) and the number who could be exposed to the situation under which the software could be used

**Likelihood of the risky situation arising** – likelihood of the risky situation arising when the system is used in the care of a patient with the possible condition (e.g., cancer, hospital admission, subject of a procedure)

**Transparency of software operation, data, and knowledge content sources** – visibility of the data, algorithms, and knowledge sources being used in the generation of the system’s output

* The consumer of the system output could be a human user directly, or could be another system
  + On one end of the spectrum, the recipient of the system output can view all the data, algorithms, and knowledge content used to generate the system output
  + On the other end of the spectrum, the system could be operating as a black box

**Ability to mitigate harmful condition** – ability for a human to detect and take action to mitigate any potential for harm

* Human intermediary could be mandatory (i.e., system output goes directly to a human), optional, or excluded (closed-loop operation)

**Complexity of software and its maintenance** – complexity of the system software and the process for updating/maintaining it

* Software may be complex, but can be tested comprehensively and can be operated reliably; complexity is considered in the risk assessment, but does not determine the level of risk alone

**Complexity of implementation and upgrades** – complexity of human effort required to implement the software and it upgrade

* Having a lot of “build” flexibility can allow helpful customization of the usability and effectiveness of the software, but it can provide many avenues for introducing risky situations not present in the “vanilla” system

**Complexity of training and use** – complexity of learning to use the software effectively and reliably

* Proxy for this is number of hours required for training

**Use as part of more comprehensive software/hardware system** – the anticipated use as a part of a broader software system

* Likely considerations could include:
  + Typical number of interfaces
  + Whether interfaces use mature, broadly adopted content and messaging standards
  + Level of redundancy to avoid single point of failure
  + Clarity of interpretation of system output

**Network connectivity**