
Tufts Security & Privacy Lab

Tufts UNIVERSITY

SWARTHMORE

Northeastern University

UMBC

# The Threat Modeling Naturally Tool

## An Interactive Tool Supporting More Natural Flexible and Ad-Hoc Threat Modeling

Ronald E. Thompson
Christopher Pellegrini
Madison Red
Esam Nesru
Richard Zhang
Mira Jain
Yaejie Kwon
Caroline Chin
Lisa Dang
Daniel Votipka

**WSIW 24**

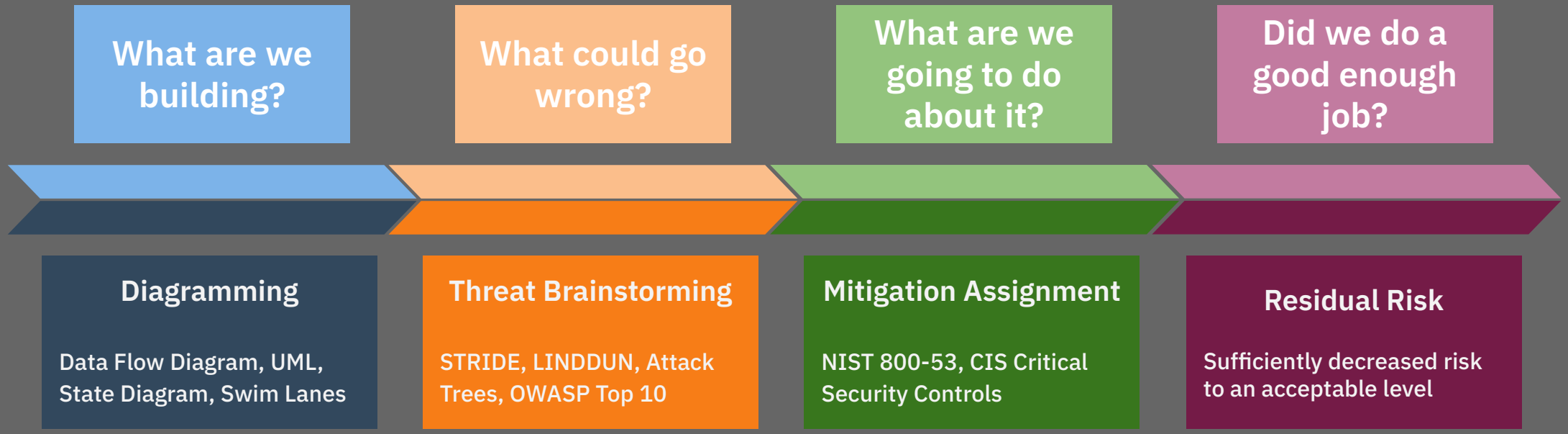
# Agenda

Threat Modeling in Practice

Current Tool Landscape

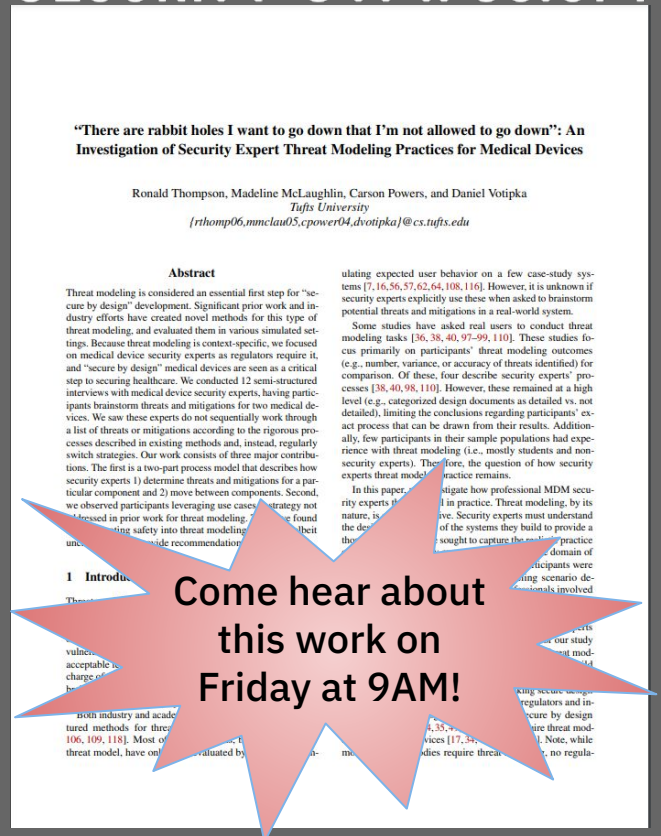
TMNT
- Design Goals
- Implementation
- Use Cases

# Threat modeling is a structured process to evaluate security at an architectural level




*4 Questions from Adam Shostack & Threat Modeling Manifesto*

# 33RD USENIX SECURITY SYMPOSIUM



"There are rabbit holes I want to go down that I'm not allowed to go down": An Investigation of Security Expert Threat Modeling Practices for Medical Devices

Ronald Thompson, Madeline McLaughlin, Carson Powers, and Daniel Votipka
*Tufts University*
*{rthomp06,mmclau05,cpower04,dvotipka}@cs.tufts.edu*

**Abstract**

Threat modeling is considered an essential first step for "secure by design" development. Significant prior work and industry efforts have created novel methods for this type of threat modeling, and evaluated them in various simulated settings. Because threat modeling is context-specific, we focused on medical device security experts as regulators require it, and "secure by design" medical devices are seen as a critical step to securing healthcare. We conducted 12 semi-structured interviews with medical device security experts, having participants brainstorm threats and mitigations for two medical devices. We saw these experts do not sequentially work through a list of threats or mitigations according to the rigorous processes described in existing methods and, instead, regularly switch strategies. Our work consists of three major contributions. The first is a two-part process model that describes how security experts 1) determine threats and mitigations for a particular component and 2) move between components. Second, we observed participants leveraging use cases [illegible] strategy not addressed in prior work for threat modeling. [illegible] found [illegible] safety into threat modeling [illegible] side recommendation [illegible]

ulating expected user behavior on a few case-study systems [7,16,56,57,62,64,108,116]. However, it is unknown if security experts explicitly use these when asked to brainstorm potential threats and mitigations in a real-world system.

Some studies have asked real users to conduct threat modeling tasks [36, 38, 40, 97–99, 110]. These studies focus primarily on participants' threat modeling outcomes (e.g., number, variance, or accuracy of threats identified) for comparison. Of these, four describe security experts' processes [38, 40, 98, 110]. However, these remained at a high level (e.g., categorized design documents as detailed vs. not detailed), limiting the conclusions regarding participants' exact process that can be drawn from their results. Additionally, few participants in their sample populations had experience with threat modeling (i.e., mostly students and non-security experts). Therefore, the question of how security experts threat model [illegible] practice remains.

In this paper, [illegible] investigate how professional MDM security experts [illegible] in practice. Threat modeling, by its nature, is [illegible] ive. Security experts must understand the des[illegible] of the systems they build to provide a tho[illegible] sought to capture the [illegible] practice [illegible] domain of [illegible] ticipants were [illegible] ing scenario de- [illegible] onals involved [illegible]

1 Introdu[illegible]

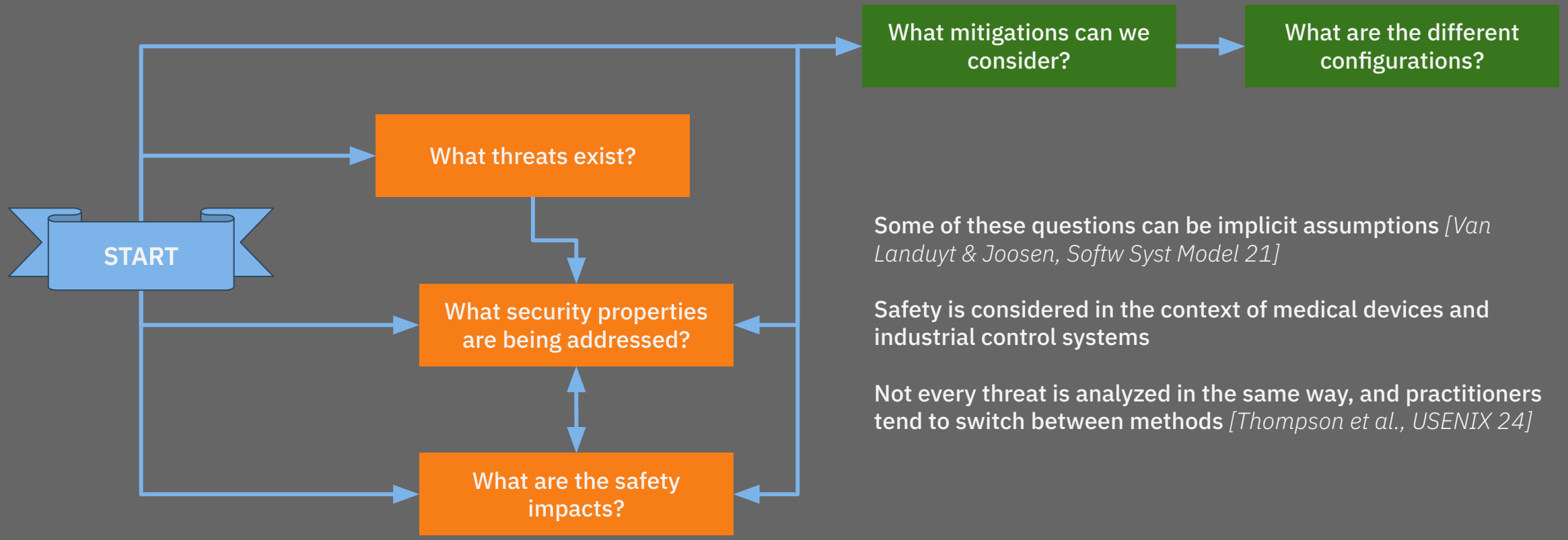
Come hear about this work on Friday at 9AM!

## We found that practitioners have more specific questions for threat brainstorming & mitigation assignment

| Diagramming | Threat Brainstorming | Mitigation Assignment | Residual Risk |
|---|---|---|---|
| | What threats exist? | What mitigations can we consider? | |
| | What security properties are being addressed? | What are the different configurations? | |
| | What are the safety impacts? | | |

*Thompson et al. There are rabbit holes I want to go down that I'm not allowed to go down. USENIX Security 24*

# Practitioners don't address these questions linearly



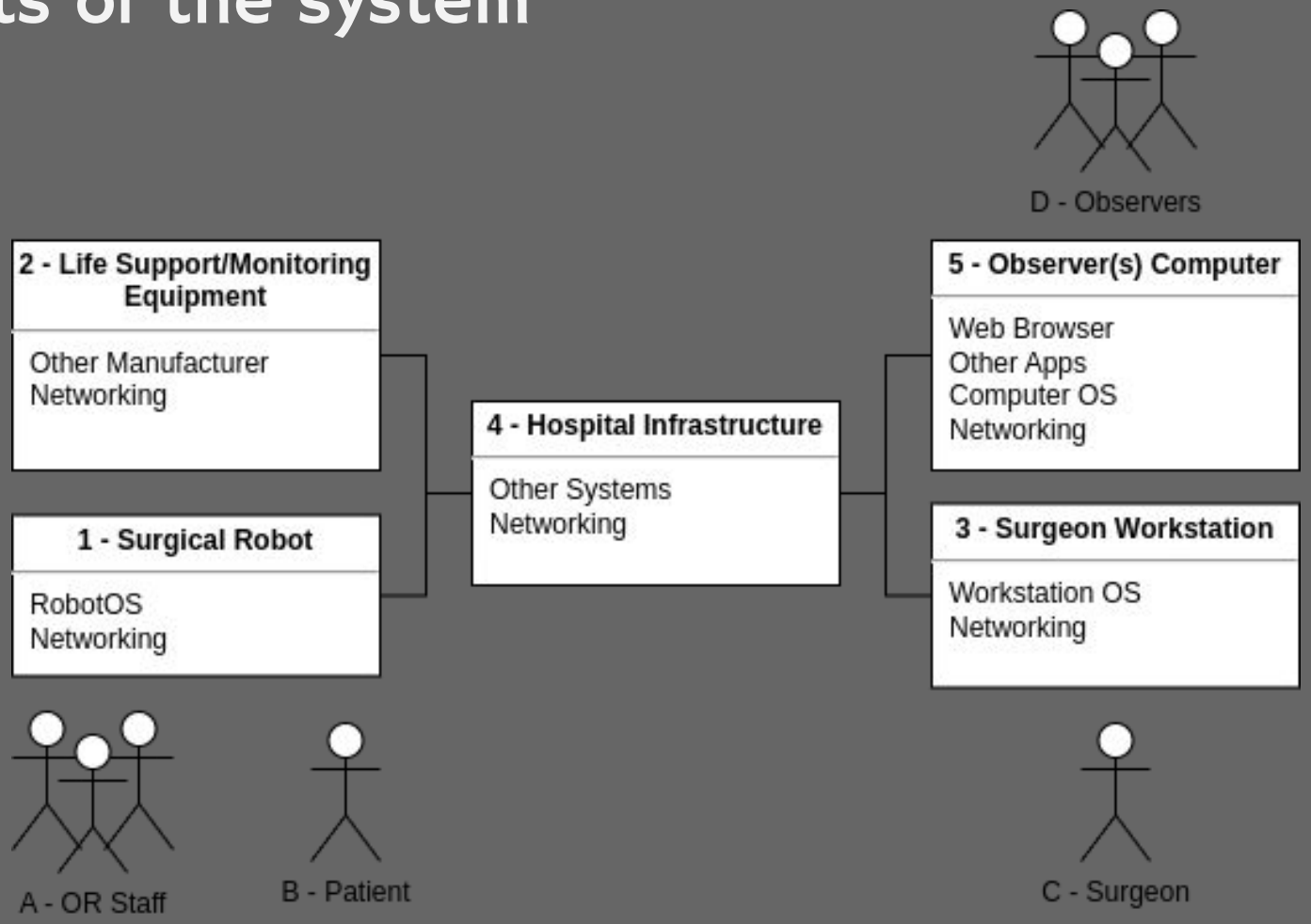

Some of these questions can be implicit assumptions *[Van Landuyt & Joosen, Softw Syst Model 21]*

Safety is considered in the context of medical devices and industrial control systems

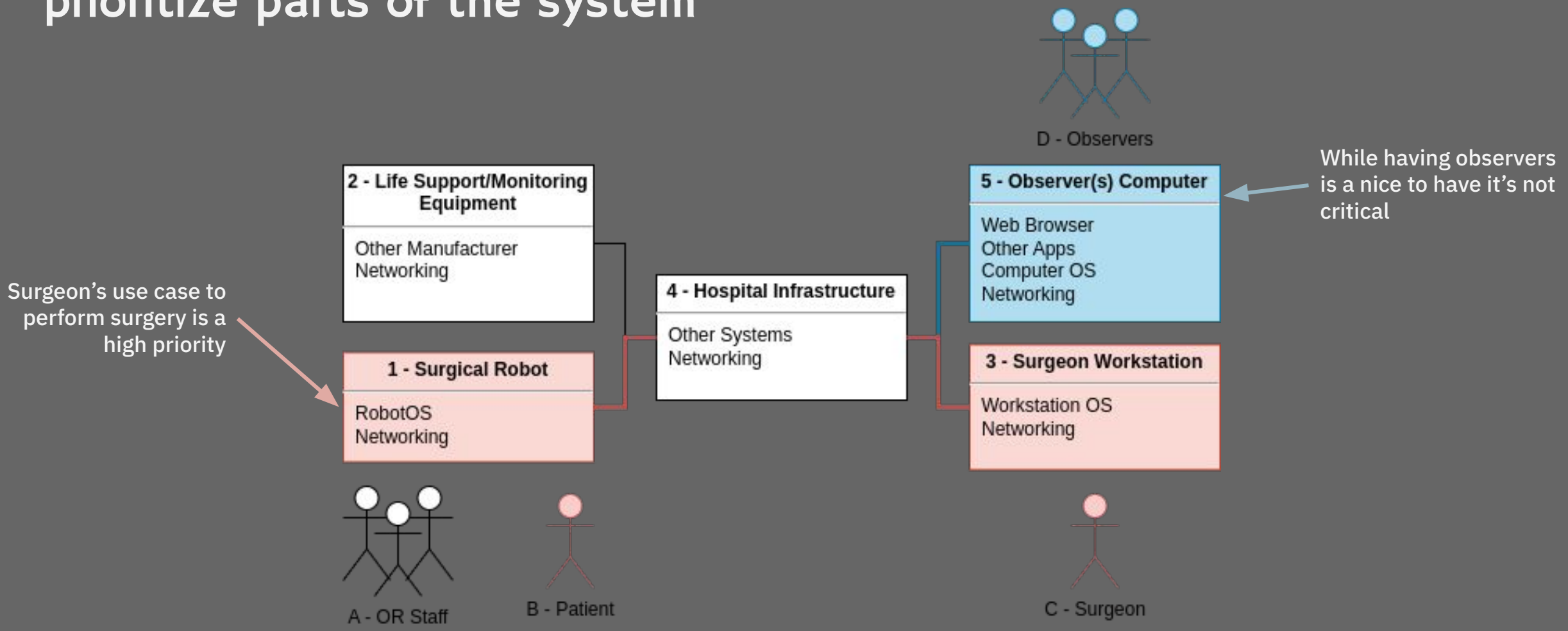
Not every threat is analyzed in the same way, and practitioners tend to switch between methods *[Thompson et al., USENIX 24]*

*Thompson et al. There are rabbit holes I want to go down that I'm not allowed to go down. USENIX Security 24*

# Dataflows are not the only flows considered, workflows help prioritize parts of the system



# Dataflows are not the only flows considered, use-cases help prioritize parts of the system



# Tools for threat modeling vary in sophistication and usability

## Domain-Specific Languages

Meta Attack Language

threatspec


Threagile
Agile Threat Modeling

## Diagramming Tools

Visio

diagrams.net

## Applications w/ No Databases

OWASP
Threat Dragon

aws

threat-composer

## Applications w/ Threat/Mitigation Databases

Microsoft Threat
Modeling Tool

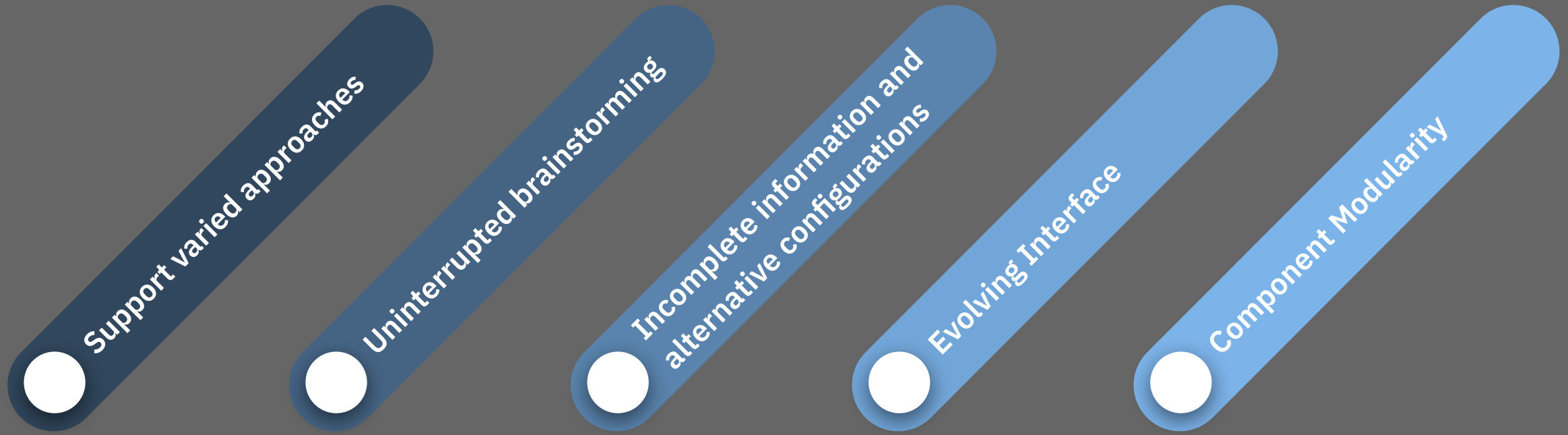
IriusRisk

ThreatModeler®

## However, they do not accommodate the “natural process” used by practitioners

- Force users to adopt approaches/assumptions of the tool designers (or provide no suggestions)
- Present threats or mitigations in separate interfaces from where the diagramming takes place
- Require users to select all the threats/mitigations that don’t apply/apply

# Five design goals that focus on usability and flexibility of process



## Support varied approaches

*Users have varied approaches to threat modeling, often adjusting their process while looking at the same system. Automation should fade into the background, suggesting threats and mitigations related to the current focus, only broadening focus when it appears the user is stuck.*

*Thompson et al. There are rabbit holes I want to go down that I'm not allowed to go down. USENIX Security 24*

Not forcing user into a prescribed method by the tool designers, but instead allow it to be defined by the user.

Includes both high level threat modeling methods (such as PASTA) and more specific brainstorming (STRIDE, LINDDUN).

## Uninterrupted brainstorming

*Various configurations may exist for a single system. Threat modeling tools should support this annotation and recommend common configurations the user may not have considered.*

*Thompson et al. There are rabbit holes I want to go down that I'm not allowed to go down. USENIX Security 24*

Provide recommendations for the part of the system that is currently being investigated and anticipate what the user might look at next

As architects are looking at their system, they may need to accommodate different configurations based on customer needs.

Instead of having to create a whole new threat model, the tool should allow them to specify this different logic.

## Incomplete information and alternative configurations

*Because there are a variety of approaches when navigating the system, tooling should allow them to cycle through alternative visualizations. This includes allowing them to isolate specific use cases, as this was common among participants.*

*Thompson et al. There are rabbit holes I want to go down that I'm not allowed to go down. USENIX Security 24*

Most tools just provide a single view as a Data Flow Diagram - which has found to not be sufficient *

Allow users to create focused views of a use case or dive more deeply into a specific part of the system

* Sion et al. Security Threat Modeling: Are Data Flow Diagrams Enough? ICSEW 20

## Evolving Interface

Interfaces with drag-and-drop, common among threat modeling interfaces are easier for experts to navigate*

Novices should be presented with specific instructions and views should be refined as they continue to learn and improve‡

* Schniederman and  Plaisant. Designing the user interface: strategies for effective human-computer interaction. 2010.

‡ Lim, Benbasat, and Todd. An experimental investigation of the interactive effects of interface style, instructions, and task familiarity on user performance. TOCHI 96

## Component Modularity

Designing modular software should promote*

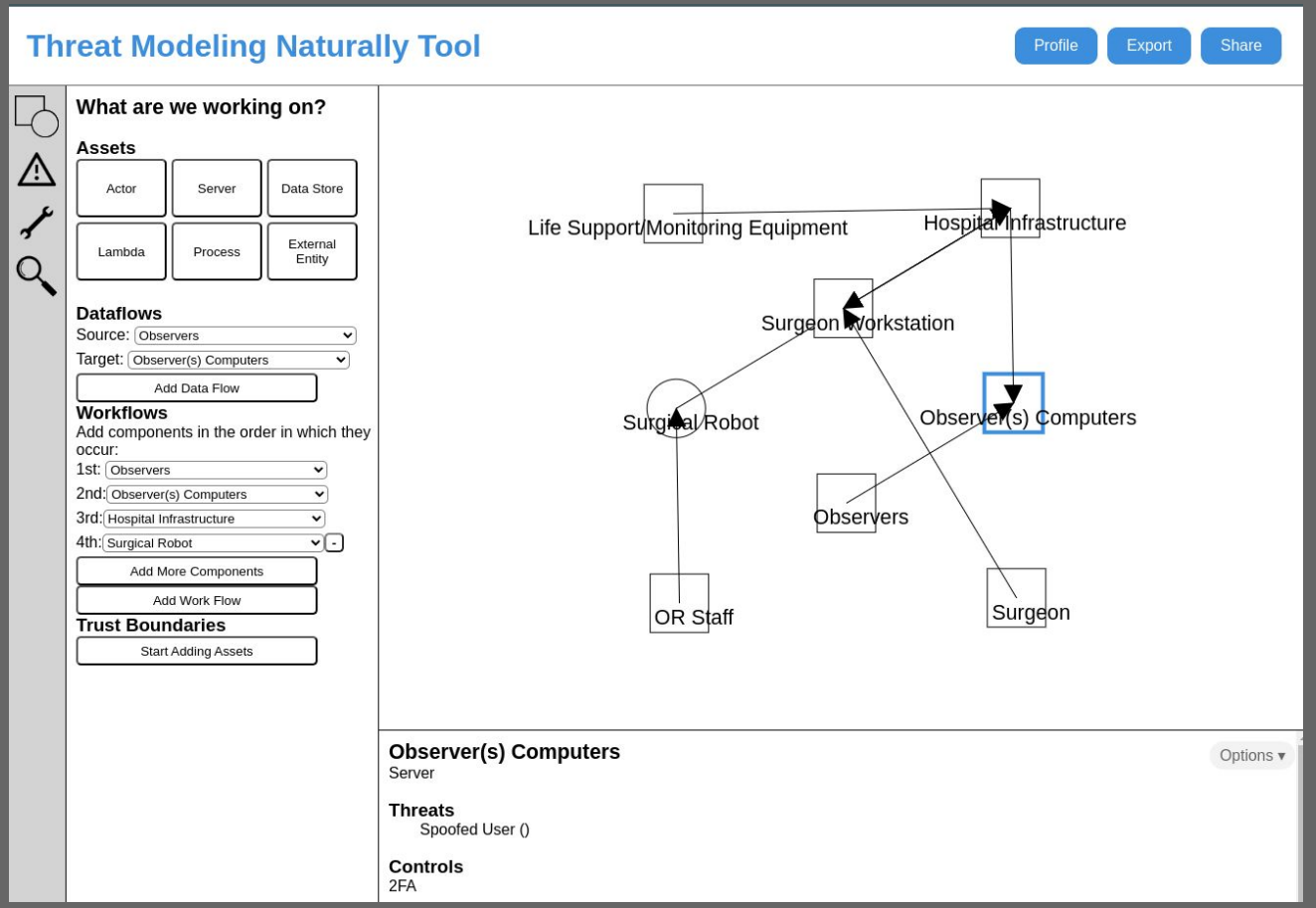
- Changeability
- Independent Development
- Comprehensibility

Ensuring that the tool is modular is part of software engineering best practice

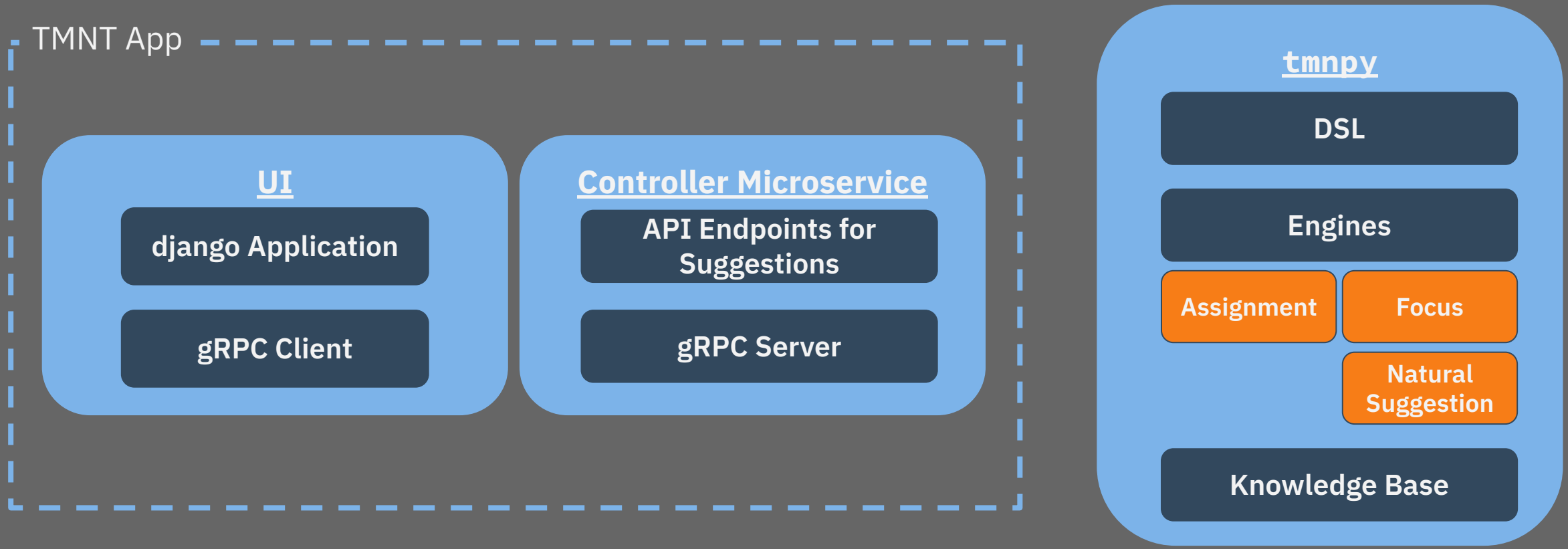
Allow for easier open source contributions

* Parnas. On the Criteria To Be Used in Decomposing Systems into Modules. Communications of the ACM. 1972.

# Introducing the Threat Modeling Naturally Tool

# We built a prototype application as well as a python package that includes the DSL, Engines, and Knowledge Base



# TMNT delivers on our design goals, we plan on continuing to develop the tool and to make it publicly available to researchers

1. **Supports varied approaches** - Our flexible DSL allows users to define a threat model in whatever way they desire
2. **Allows uninterrupted brainstorming** - Our natural suggestion engine doesn't force the user to check all of the threats in a separate workflow before matching threats to mitigations
3. **Allows for incomplete information and alternative configurations** - Our engine provides suggestions based on the information provided and updates as the user adds more detail.
4. **Has component modularity** - Our DSL is completely independent of the other parts of the system and has a flexible API allowing users to change and add definitions. The UI is not required to run the Natural Suggestion Engine.

We hope to add functionality for an interface that presents information based on user's knowledge as well as for users to be able to specify logic splits based on configuration differences

## We have put together a few different use cases for researchers, tool builders, & practitioners

Use Case 1: Researchers wanting to study different threat modeling methodologies - UI

Use Case 2: Researchers wanting to test usability with different threat / mitigation recommendations (WoZ) - UI, Controller

Use Case 3: Tool developers who want to leverage natural suggestions - tmnpy

Use Case 4: Practitioners can use it! - All parts

# TMNT is an open source project available at on Github as well as [tsp.cs.tufts.edu/tmnt](tsp.cs.tufts.edu/tmnt)

**Summary**:

Allows researchers to conduct threat modeling user studies that is flexible and doesn't interrupt their natural process

Consists of four major parts: DSL, Knowledge Base, Suggestions Engine, and User Interface

Available to the larger community and users are encouraged to contribute to its functionality and knowledge bases

**Funding**:

CISCO Medcrypt

# Appendix

# Defining `tmnpy` as a threat modeling language

Our grammar consists of four symbols, allowing us to define threat statements in a threat model.

> An **[ISSUE]** impacts **[COMPONENT]** in our system, which leads to **[FINDING]**. This can be addressed by **[MITIGATION]**.

Each of these defines more complex symbols that allow users to define their threat model based on their requirements rather than forcing them to adopt a specific method.

# Suggestions for Issues and Mitigations defined in the TMNT Knowledge Base

Issues & Mitigations are defined with associated Component types - e.g. Datastores

Each Issue [or Mitigation] can be associated with sibling Issue(s) [Mitigations(s)], parent(s), and children

Similar to how current knowledge bases are structured such as MITRE's Common Weakness Enumeration (CWE) & Common Attack Pattern Enumerations and Classifications (CAPEC)


Relying on Open Source model to encourage contributions of additional knowledge bases and improving those currently defined

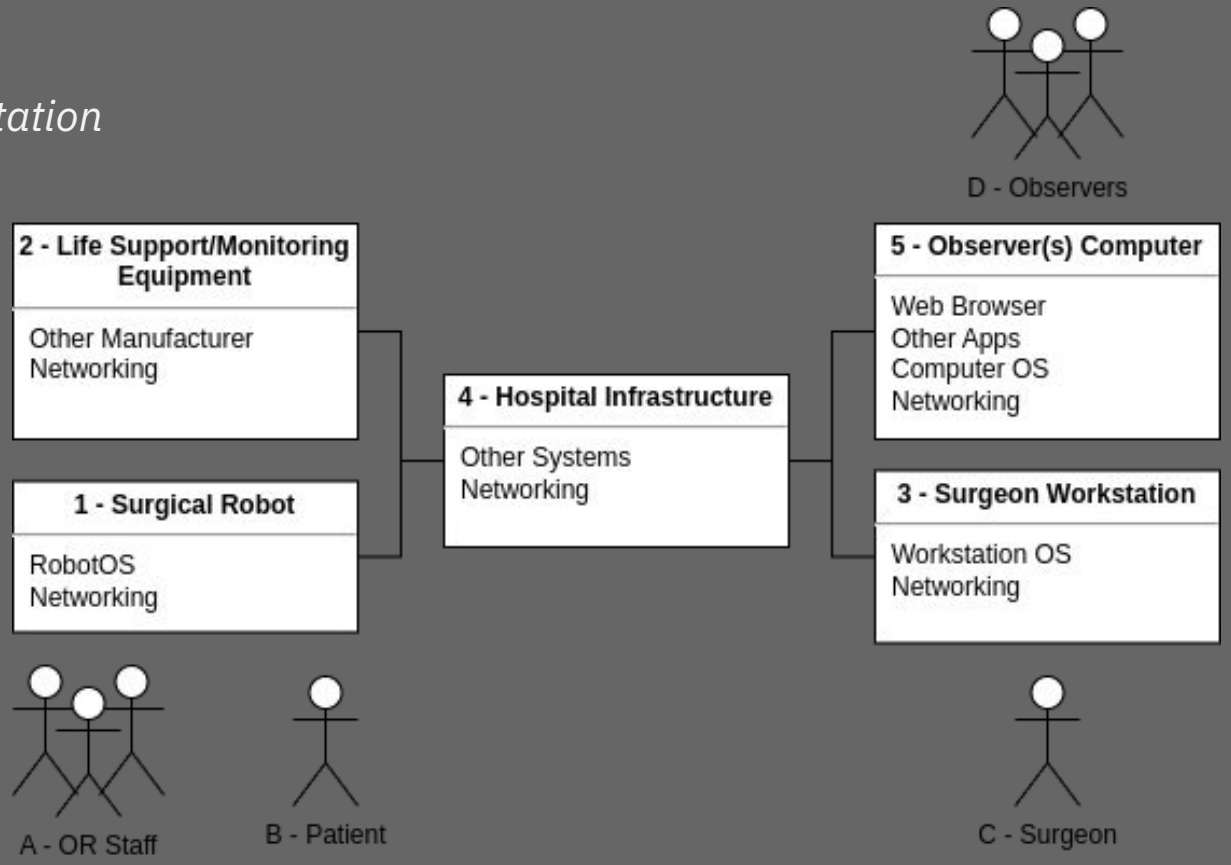
# Turning Suggestions into “Natural” Suggestions

As users create their threat model, adding components and specifying threats/mitigations, these actions are used to tailor suggestions to be based on what they are currently doing

*NOTE: We have implemented parts of this functionality and not all of it at this point*

# Turning Suggestions into "Natural" Suggestions - Example

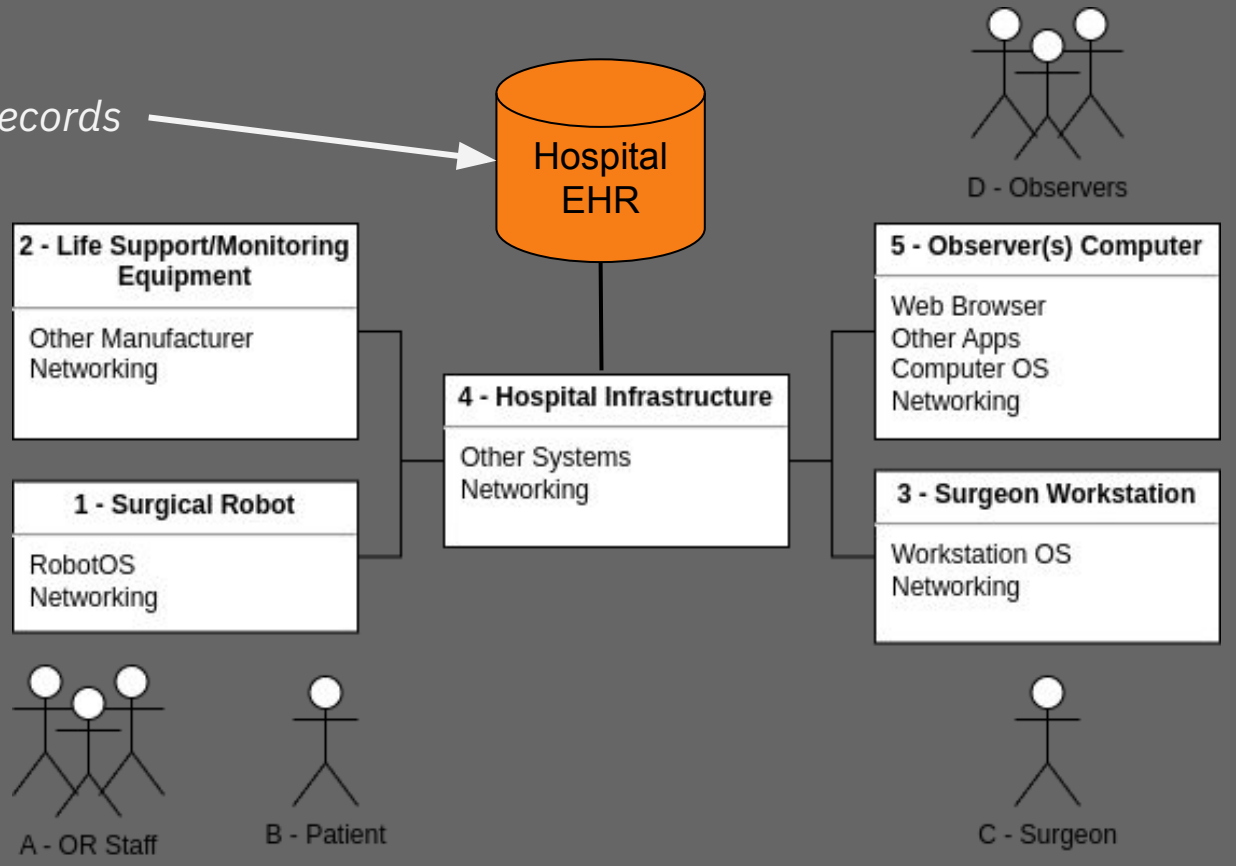
*User has generated the following system representation*




*NOTE: We have implemented parts of this functionality and not all of it at this point*

# Turning Suggestions into "Natural" Suggestions - Example

*User adds a new datastore for Electronic Health Records*

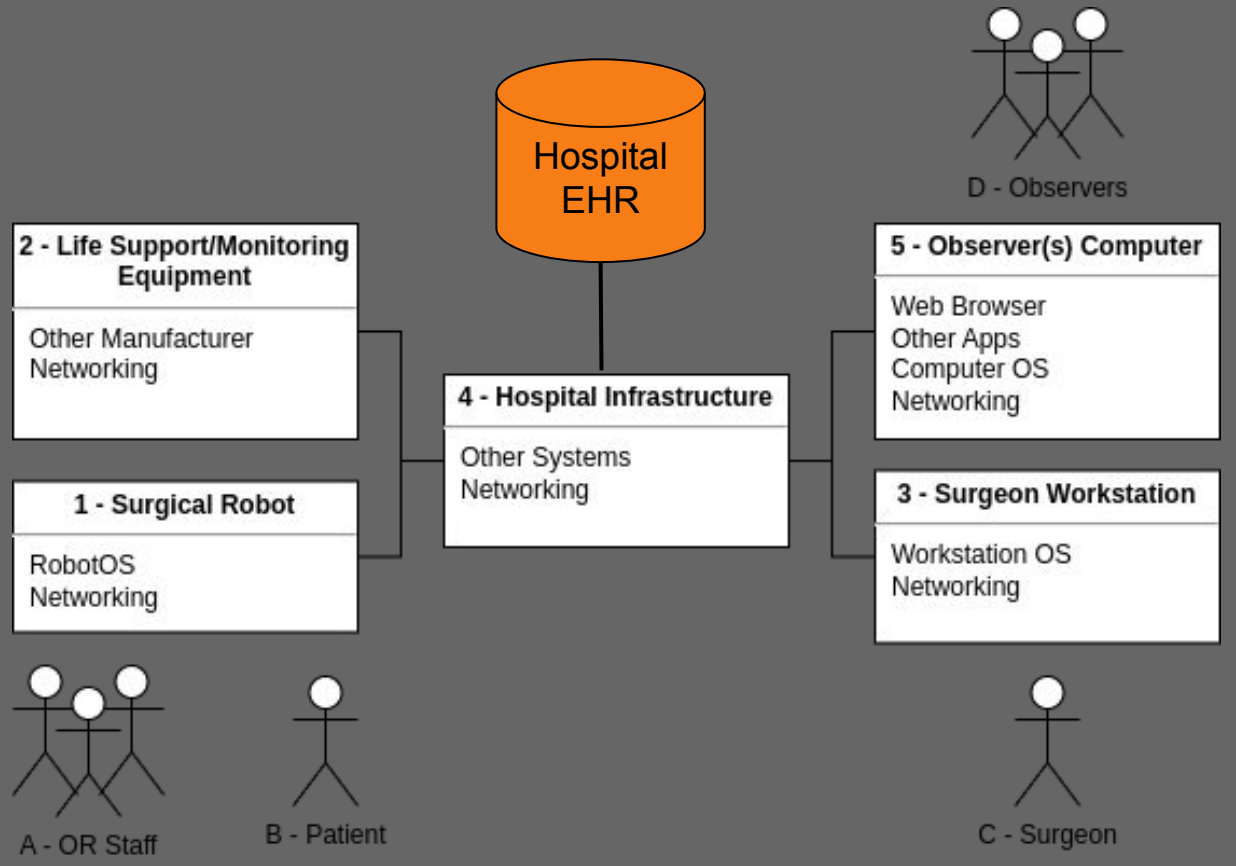



*NOTE: We have implemented parts of this functionality and not all of it at this point*

# Turning Suggestions into "Natural" Suggestions - Example

*Natural Suggestion Engine asks...*

- Do you encrypt this data?
- What would happen if a threat actor tampered with this data?



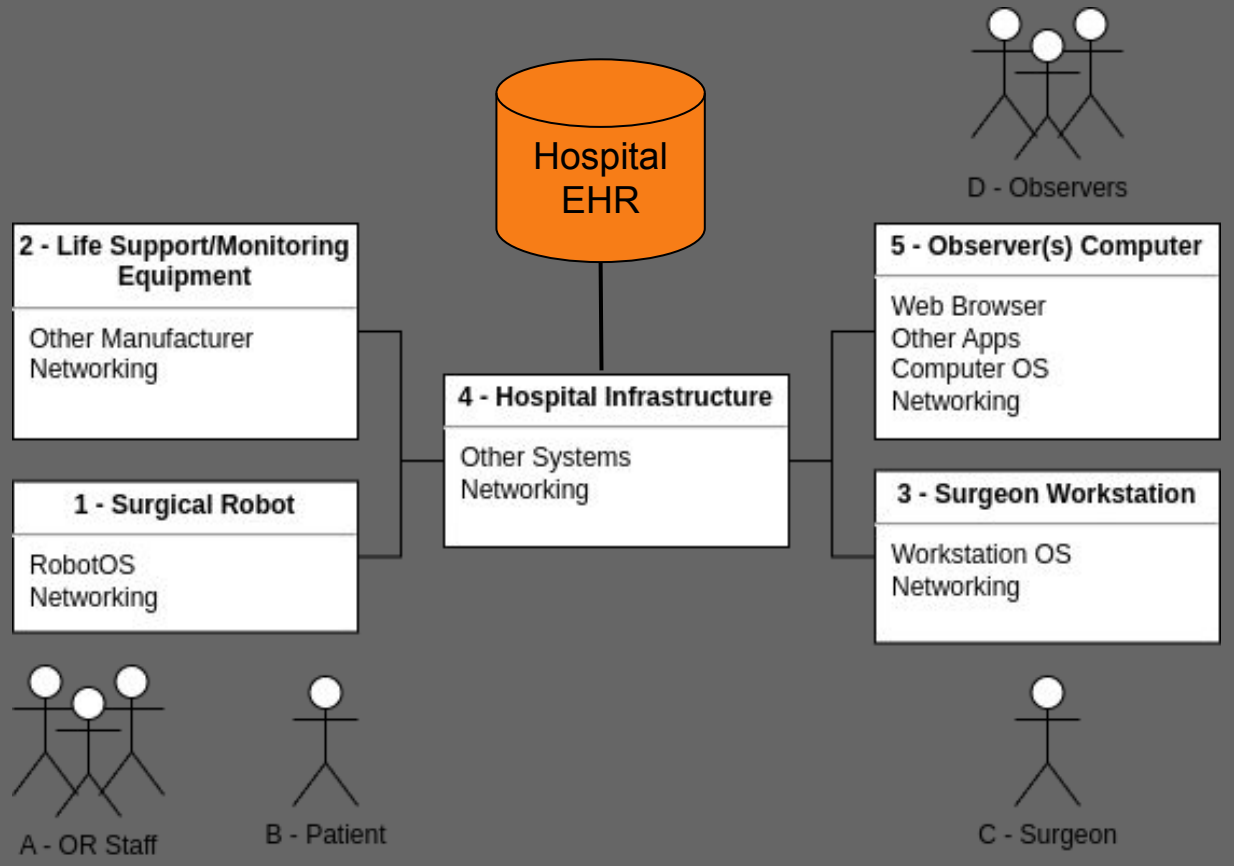

*NOTE: We have implemented parts of this functionality and not all of it at this point*

# Turning Suggestions into "Natural" Suggestions - Example

*Natural Suggestion Engine asks...*

- Do you encrypt this data?
- What would happen if a threat actor tampered with this data?

*User responds "Yes"*




*NOTE: We have implemented parts of this functionality and not all of it at this point*

# Turning Suggestions into "Natural" Suggestions - Example

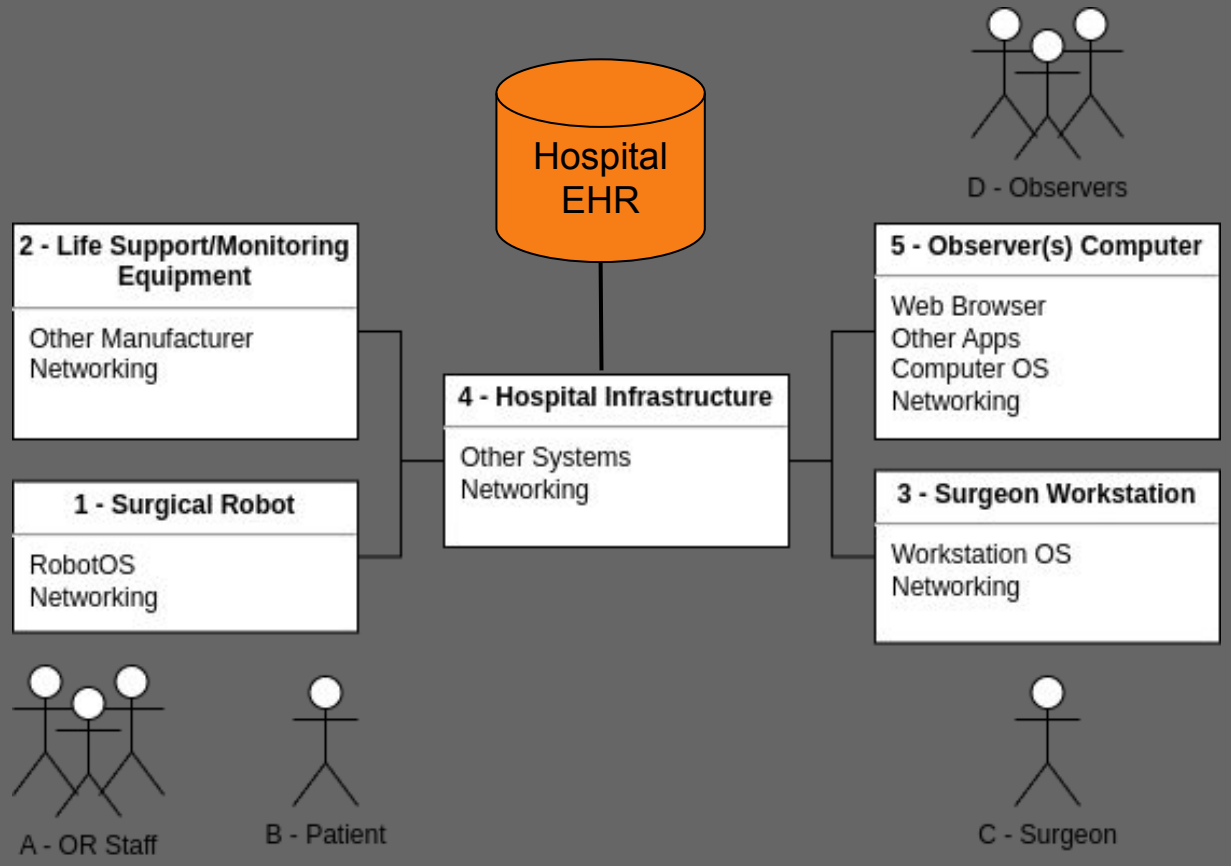
*Natural Suggestion Engine asks...*

- Do you encrypt this data?
- What would happen if a threat actor tampered with this data?

*User responds "Yes"*

*New suggestions from the Engine...*

- Is there anywhere else you are encrypting data?
- What would happen if a threat actor tampered with this data?




*NOTE: We have implemented parts of this functionality and not all of it at this point*