THREAT MODELLING

ROLLS ROYCE

LUKE THOMAS

N.LUKE.THOMAS@ROLLS-ROYCE.COM

AAYUSH MAILARPWAR

COOPER ZURANSKI

JOE SWINEY

KRISTOPHER BRIGHT

11/18/2022

MOTIVATION

- THE GOAL IS TO CREATE AWARENESS ABOUT THE POSSIBLE THREATS
- RECENTLY MORE AND MORE DEVICES HAVE BECOME INTERCONNECTED
- SECURITY IS MAINLY AN AFTERTHOUGHT
- THERE ARE VERY FEW SECURITY EXPERTS
- GIVE SOMEONE WITH NO BACKGROUND IN SECURITY THE ABILITY TO IDENTIFY VULNERABILITIES

BACKGROUND

- THREAT MODELING IS USED TO ANALYZE, DISCOVER AND DESCRIBE VULNERABILITIES IN SYSTEMS
- THREAT MODELING DESCRIBES WHAT IS BEING MODELED IN MORE CONTEXT THAN JUST SECURITY FIRST.
- THREAT MODELING CAN BE USED FOR SECURITY SYSTEMS, AIRCRAFT, AUTOMOBILES, EMBEDDED CONTROLS, PHONES, CAMERAS, WIRELESS EQUIPMENT, AND EVEN THINGS AS SIMPLE AS PHYSICAL ROOMS.

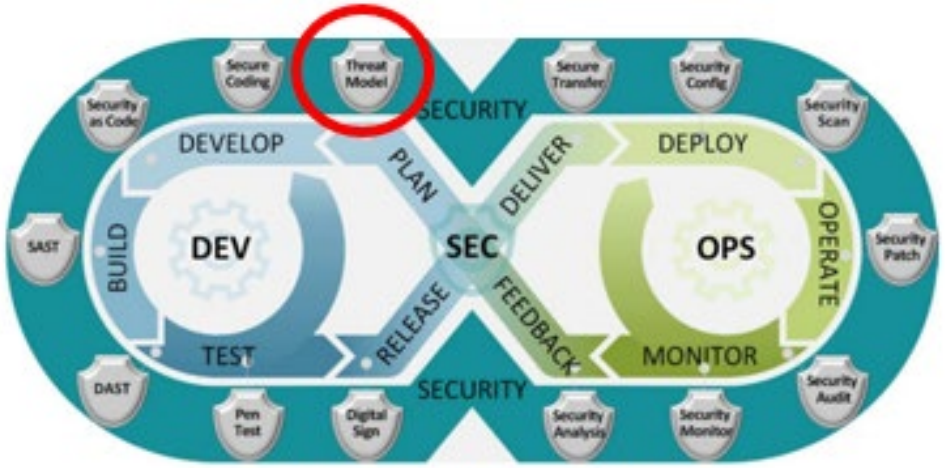
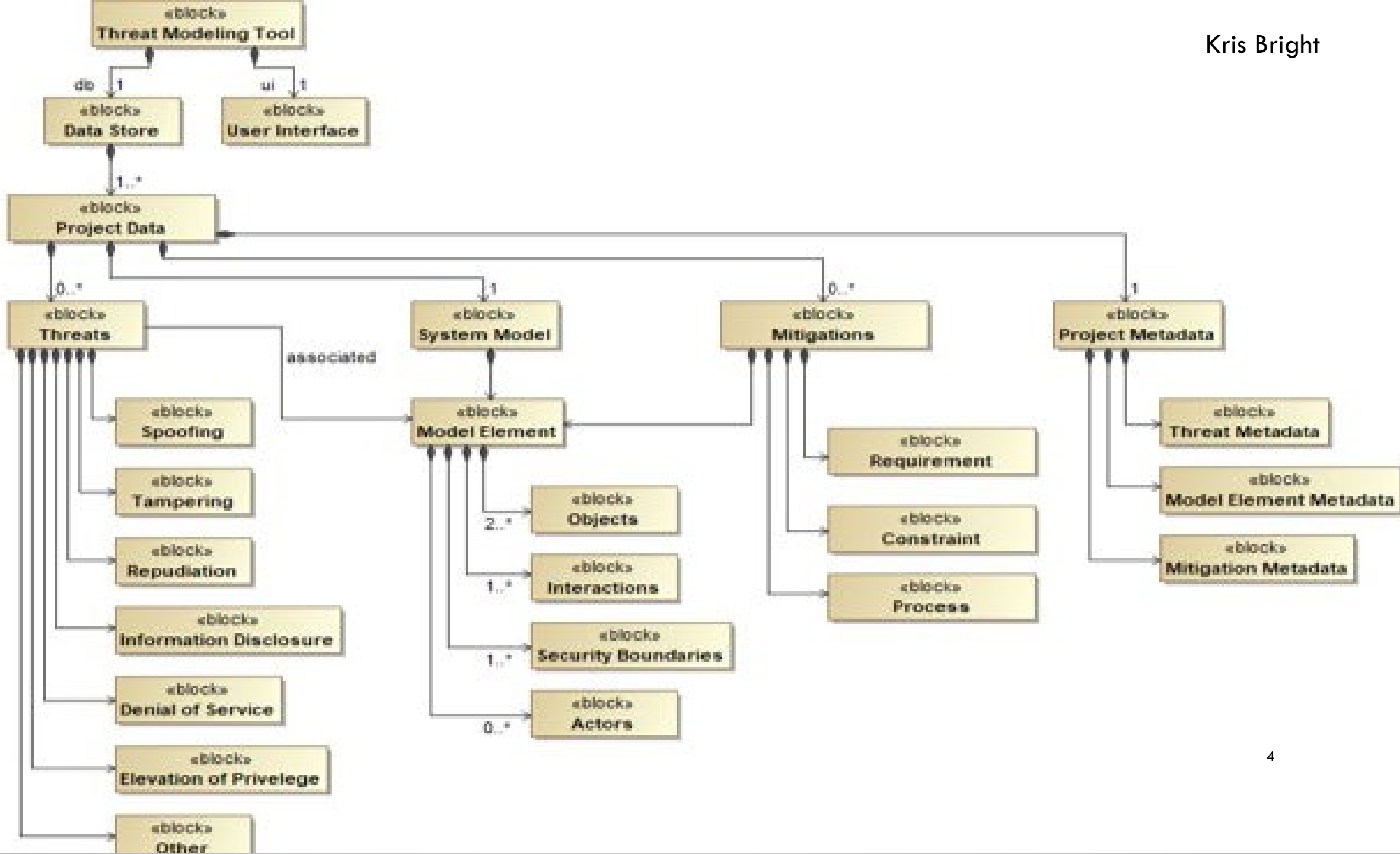


Figure 3: DevSecOps Software Lifecycle


«block»
Threat Modeling Tool
db
1
ui
1
«block»
Data Store
«block»
User Interface
1..*
«block»
Project Data
0..*
1
0..*
1
«block»
Threats
«block»
System Model
«block»
Mitigations
«block»
Project Metadata
associated
«block»
Spoofing
«block»
Tampering
«block»
Repudiation
«block»
Information Disclosure
«block»
Denial of Service
«block»
Elevation of Privelege
«block»
Other
«block»
Model Element
2..*
«block»
Objects
1..*
«block»
Interactions
1..*
«block»
Security Boundaries
0..*
«block»
Actors
«block»
Requirement
«block»
Constraint
«block»
Process
«block»
Threat Metadata
«block»
Model Element Metadata
«block»
Mitigation Metadata

PROJECT REQUIREMENTS

- TECHNICAL REQUIREMENTS
  - USER INTERFACE – EXTREMELY SIMPLE GUI (WYSIWYG)
  - ASSISTED FUNCTION – TOOLTIPS, VISUAL GUIDANCE, ON/OFF CAPABILITIES
  - IMPORT/EXPORT FUNCTIONALITY
  - SAVE AND LOAD
  - VERSION CONTROLLED – GITHUB
  - FRONTEND AND BACKEND : ANGULAR AND PYTHON
  - EXTENDABLE CODE (MINIMUM VIABLE PRODUCT)
  - MAINTAINABLE CODE

REQUIREMENTS CONTINUED

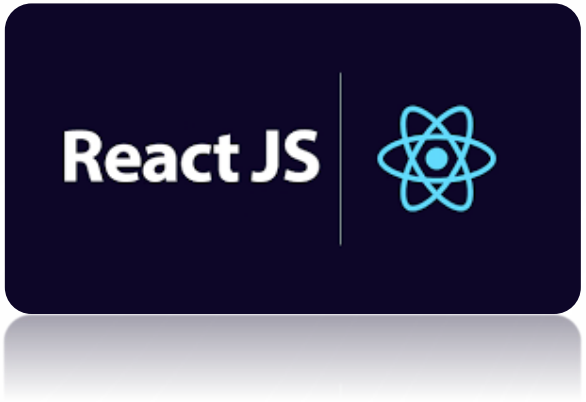
- STANDARD REQUIREMENTS
  - OPEN SOURCE
  - REGULAR CODING STANDARDS
  - PUSH/PULL REQUEST
  - USING BRANCHES ON VERSION CONTROL NOT MAIN
- CONSTRAINT REQUIREMENTS
  - NO ADMIN
  - COLORBLIND ADAPTABILITY
  - TESTABLE CODE
  - UNIX STYLE TIME RECORDING
  - CAPABILITY TO RUN ON ANY AVERAGE LAPTOP

DESIGN OPTIONS

Joe Swiney

1) FrontEnd Selection

React	
Pros:	Cons:
• Easier to learn • An abundance of online resources • Runs natively in browser	• Not a complete framework as it requires additional packages • No team member has experience



Angular	
Pros:	Cons:
• Complete framework out of the box • An abundance of online resources • Runs natively in browser • Team members have experience	• Steeper learning curve

2) BackEnd Selection

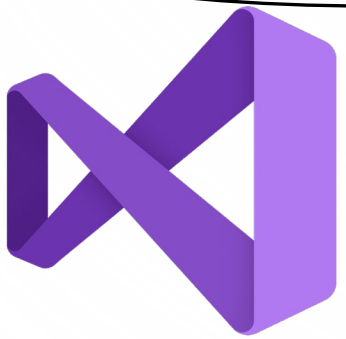
Rust	
Pros:	Cons:
• Faster execution • Similar to Java • Automatic garbage collection • Memory efficient	• No team members have used Rust • Slightly more difficult to implement • Less third-party support • Fewer online resources

Python	
Pros:	Cons:
• Three team members have used it in a professional setting • An abundance of open-source packages available • Automatic garbage collection	• Minimal learning curve for the one team member without experience • Less memory efficient • Slower execution

Joe Swiney

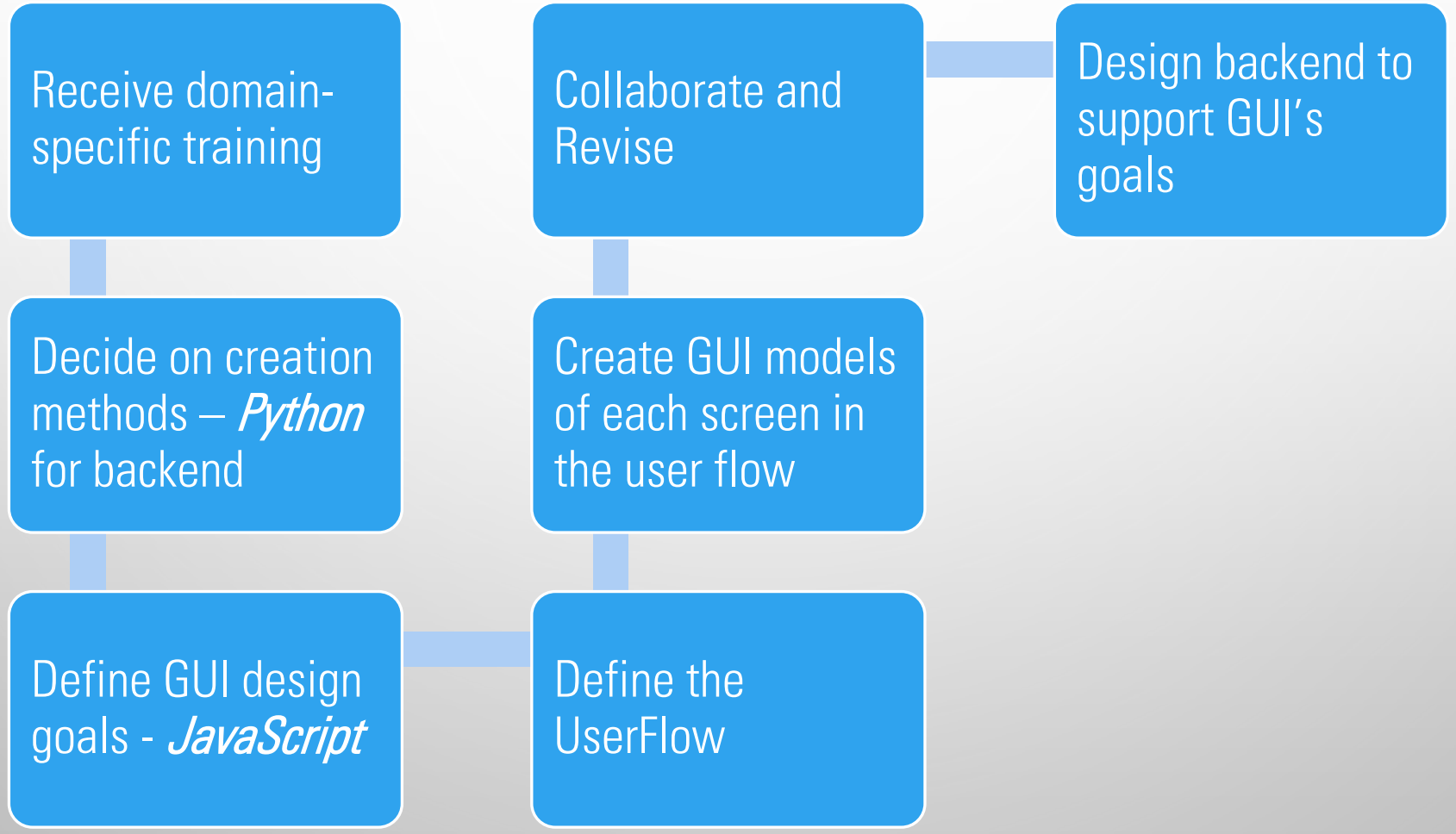
2) Integrated Development Environment Selection

JetBrains	
Pros:	Cons:
• Specialized IDEs for each language • A large variety of add-ons for code linting and syntax highlighting • Built-in, user-friendly debugger • Professional versions available free for students • Built-in GitHub integration • Optimized for Mac and PC • Recommended by CS professor for JavaScript development	• Requires a different application for each language



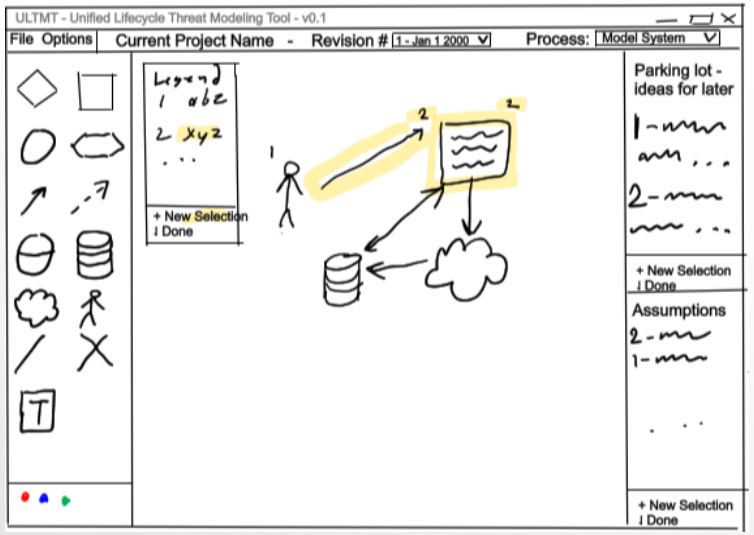
Visual Studio	
Pros:	Cons:
• Built-in, user-friendly debugger • Add-ons for all necessary languages • Both languages can be written in the same application	• Initial setup can be difficult • Not optimized for Mac • Can be slow

DESIGN TRACK

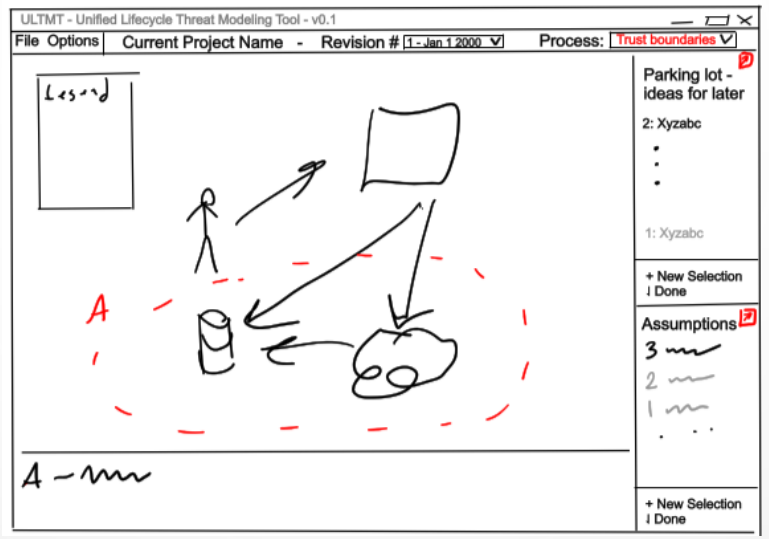


DESIGN DETAILS

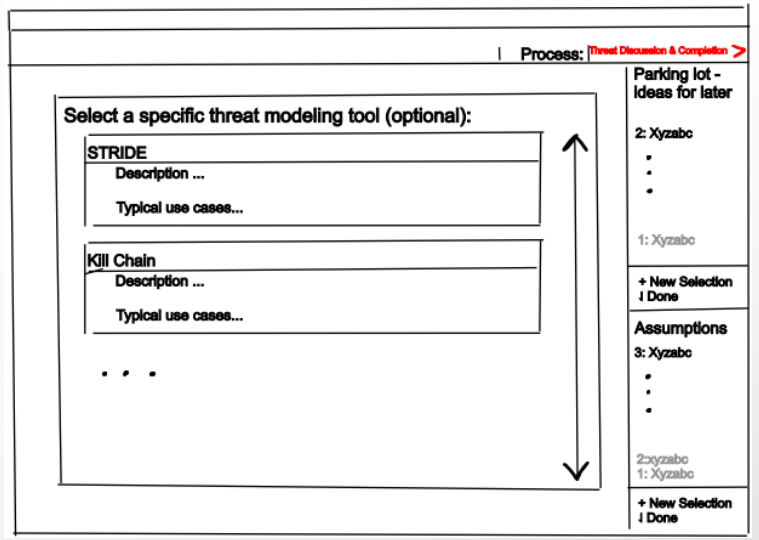
Cooper Zuranski



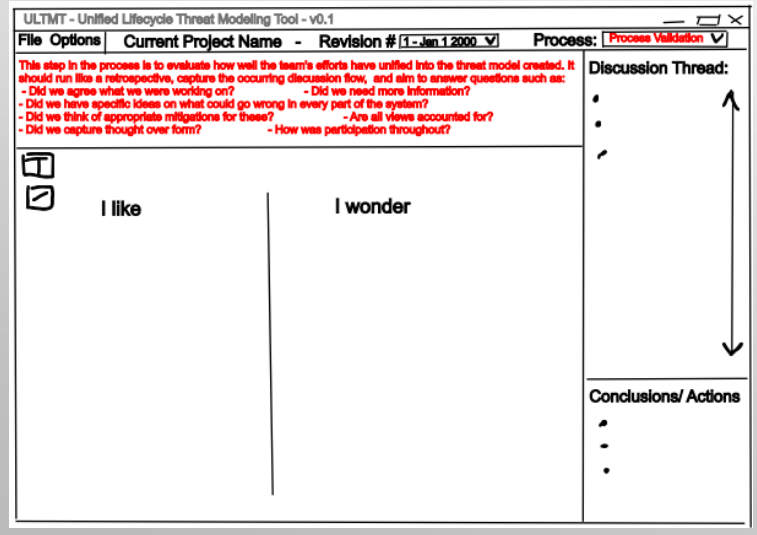
Model System



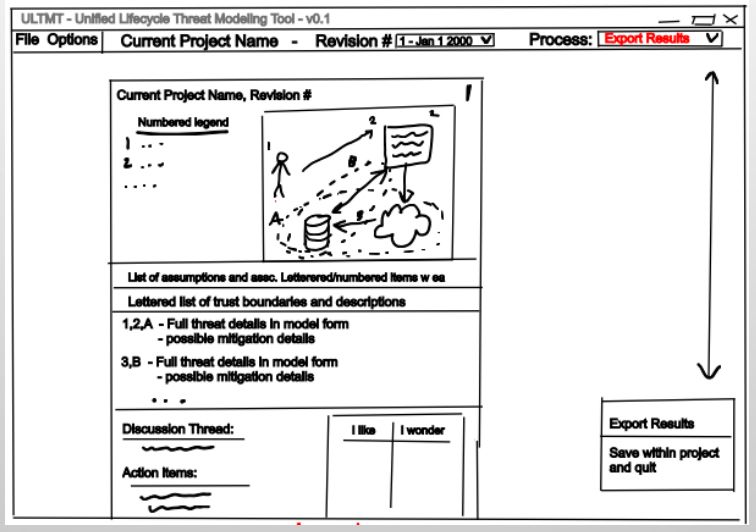
Trust Boundaries



Threat Discussion and Completion



Process Validation



Export Results

Model Confirmation

SYSTEM TEST PLAN

Kris Bright

Mocha/Chai	
Pros:	Cons:
• **Automated testing process** • **Can continuously test or perform tests on-demand** • **Can test complex functionality** • **Human-readable syntax** • **Free** • **One team member has experience, though the experience is limited**	• Complicated syntax can be confusing for newer users • Documentation isn't always clear

JEST	
Pros:	Cons:
• **Simple syntax that is human-readable** • **Good documentation available online** • **Can test complex functionality**	• No group member has experience

unittest	
Pros:	Cons:
• **Automated testing process**	• No external plug-ins

PyTest	
Pros:	Cons:
• **Automated testing process**	• Over 800 external plug-ins to speed up creation of test cases

POTENTIAL RISK AND MITIGATION

- LITTLE TO NO PHYSICAL RISK DUE AS IT'S PURELY SOFTWARE
- FAILING TO MAKE SOFTWARE EASILY EXTENDABLE
- HIDDEN SOFTWARE BUGS
- ISSUES WITH VERSION CONTROL
- FAILING TO MEET BASIC CONSTRAINTS

PROJECT PLAN

Determine Requirements	Design Initial Iteration	Threat Modeling Training	Modify Initial Iteration Design	Finalize Design	Sponsor Check In	Initial UI Implementation	UI Demonstration
Weeks 1-5							
	Weeks 5-7						
		Week 8					
			Weeks 9-11				
				Weeks 11-13			
						Weeks 13-15	
							Week 16

Fig. Semester 1 Project Plan

Integrate Backend	Write Tests	Sponsor Check In	Integration Testing	Sponsor Check In	Systems Testing	Project Refinements
Weeks 17-20						
	Weeks 20-22					
			Weeks 23-25			
					Weeks 26-29	
						Weeks 29-32

Fig. Semester 2 Project Plan

QUESTIONS?