34th Annual INCOSE international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

Automating Rule-Checking to Identify SysML Modeling Errors: A Preliminary Study in a Classroom Environment

Michael Vinarcik, Director, Digital Architecture and Requirements, SAIC

Abstract

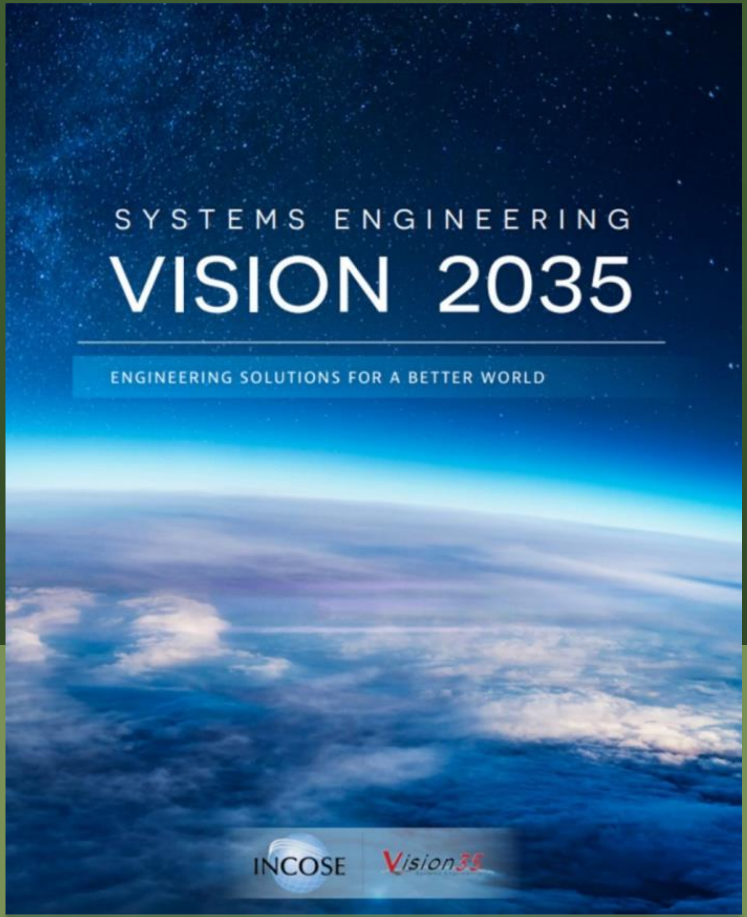
Among other benefits, the adoption of Model-Based Systems Engineering (MBSE) or Digital Engineering (DE) is expected to decrease the number of escaped errors in a project by enabling early error detection within the modeling environment. This contributes to reducing the costs associated with corrective activities, which are generally greater in later phases of development.

This paper provides an empirical insight into error detection through a study of models developed by students in a graduate MBSE course, where they leveraged the use of automated rule checking within the modeling tool. The dataset covers 10 editions of the course, spanning 2016-2023, and contains 601 models. The study shows that the term project models resulted in nearly zero latent errors when non-stylistic rules are concerned, with most of the latent errors categorized as stylistic rather than fundamental violations.

Contents

Introduction

INCOSE SE Vision 2035: Modelers Needed



- Achieving SE Vision 2035 will require capable tools, methods, and practitioners.
- Practitioners must master outcome-focused systems engineering skills, modeling languages, tools, and methods.
- Efficient and effective training is needed to upskill the individuals who will realize this vision.

MENG 5925

(SysML Modeling) Course Background

- Students in the SysML course at the University of Detroit Mercy are required to construct descriptive architecture models as homework (typically household appliances); term projects consist of larger-scale, team modeling efforts (often space probes or systems of similar size and complexity).
- A consistent (although evolving in detail) modeling style was taught throughout the period of this study.
- Automated validation rules were introduced in the Fall of 2019; these enabled grading efforts to focus more on model content and less on style and completeness.
- The rules continued to evolve over the study period to drive more consistency and completeness; they also expanded to include some minor customizations to fully synchronize behavioral and structural elements.

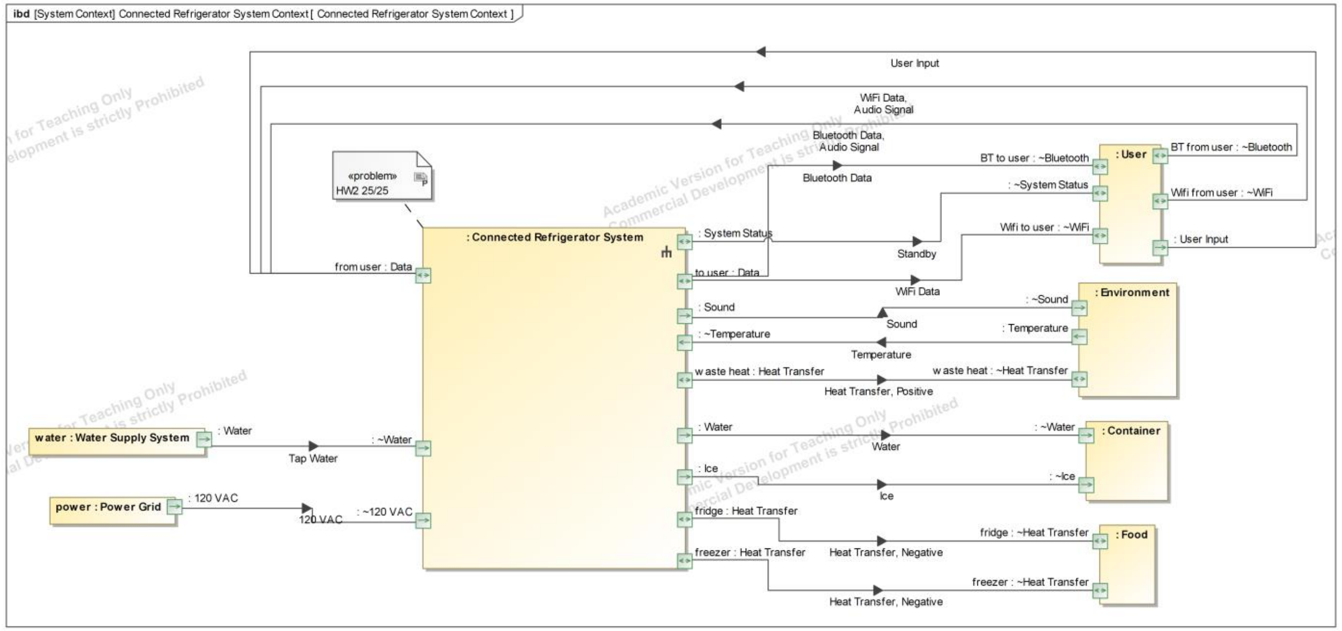
Pedagogy

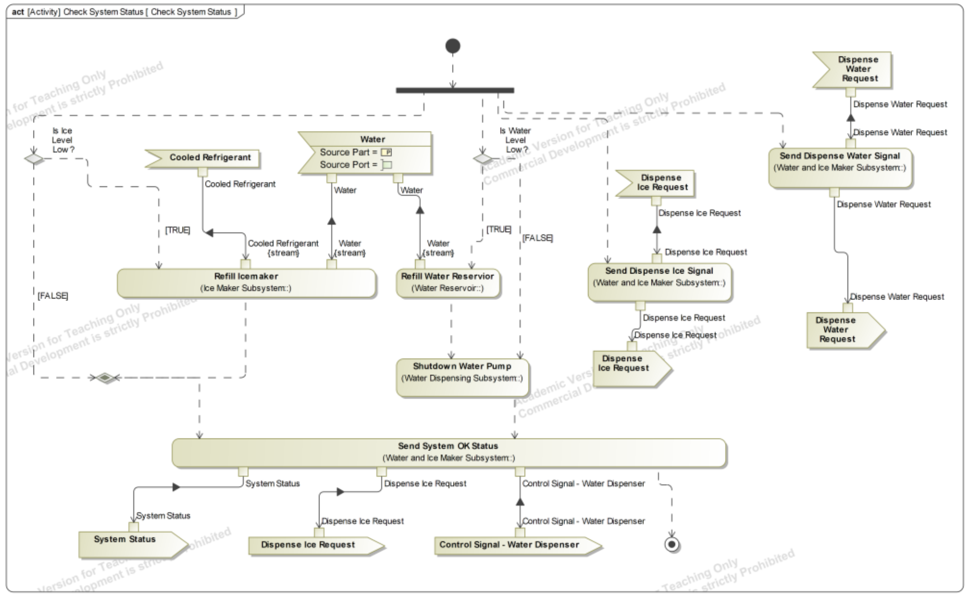
- The first half of the term focuses on individual homework assignments to teach SysML and tool fundamentals.
- The lessons are organized in a structure-first sequence (driven by operation-focused behavioral modeling).
- The latter portion of the class is focused on term projects actively driven by weekly instructor reviews.
- Teams are required to submit regular essays discussing their process, a final presentation showcasing their architecture, and reflective essays.
- A common error log is used to capture video gaffes/omissions to allow the course to be improved.

From *Forged in Fire: Teaching the Craft of Model-Based Systems Engineering, Vinarcik, 2023 INCOSE International Symposium*

Homework Outcomes

- Individual
- Model a simple system
- Familiarity with tool UI
- Understand SysML
- Construct basic structural and behavioral diagrams

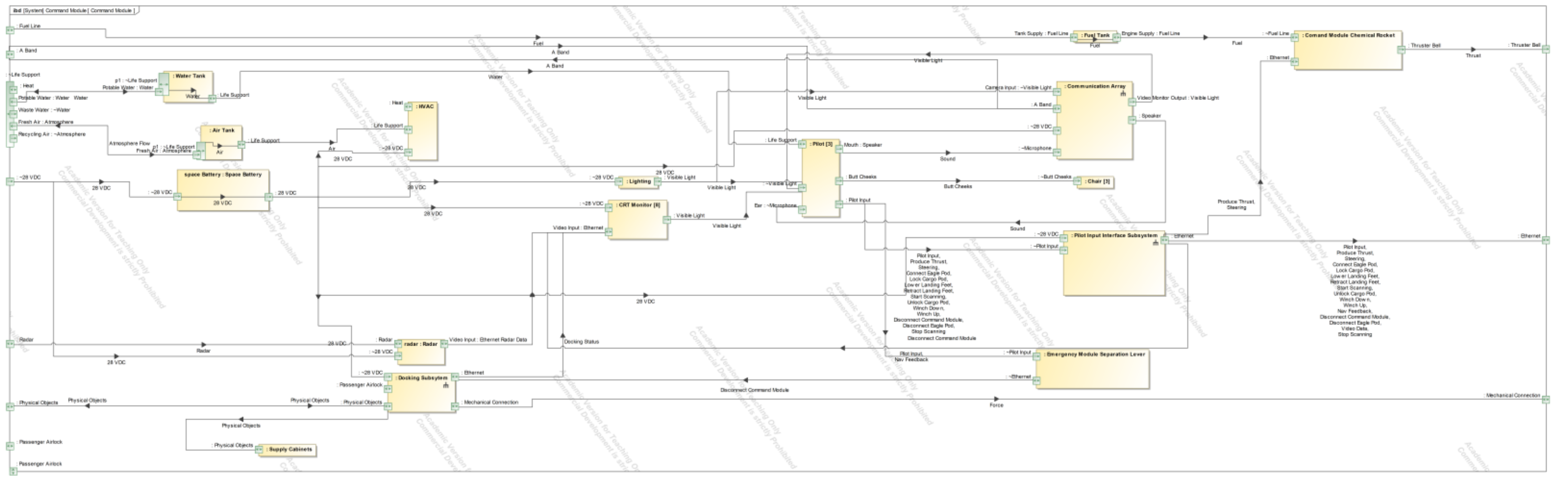




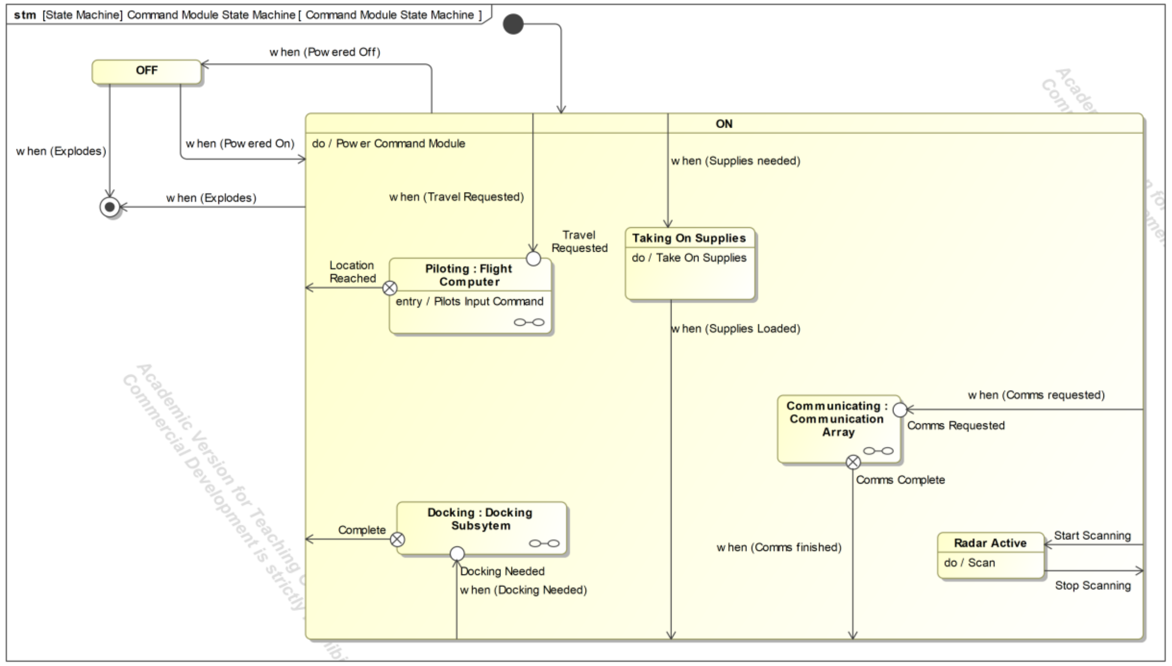
act [Activity] Check System Status [Check System Status]
Is Ice Level Low ?
Is Water Level Low ?
[TRUE]
[FALSE]
Cooled Refrigerant
Water
Source Part =
Source Port =
Cooled Refrigerant {stream}
Water {stream}
Refill Icemaker (Ice Maker Subsystem::)
Refill Water Reservior (Water Reservoir::)
Shutdown Water Pump (Water Dispensing Subsystem::)
Dispense Ice Request
Send Dispense Ice Signal (Water and Ice Maker Subsystem::)
Dispense Water Request
Send Dispense Water Signal (Water and Ice Maker Subsystem::)
Send System OK Status (Water and Ice Maker Subsystem::)
System Status
Control Signal - Water Dispenser

Project Outcomes

- Team (4-6 students)
- Model a more complicated system
- Master working as part of a larger development team



ibd [System] Command Module [Command Module]
: Water Tank
: HVAC
: Air Tank
space Battery : Space Battery
: Lighting
: CRT Monitor [4]
: Pilot [3]
: Communication Array
: Chair [3]
: Pilot Input Interface Subsystem
: Emergency Module Separation Lever
radar : Radar
: Docking Subsytem
: Supply Cabinets
: Fuel Tank
: Comand Module Chemical Rocket



Typical Project Deliverables

- Capabilities and mission threads traced to source content
- Refined capabilities and mission threads
- Initial logical architecture
- Intermediate logical architecture with internal block diagrams and state machines
- Fully developed architecture (expected to be zero error / zero info violations)

Time Commitments

- Weekly course session (1-1.5 hours)
- Asynchronous videos (SysML/Tool instruction)
- Weekly project tagups (45 minutes per team)
- Additional help sessions as needed

Model Grading Challenges

- Models are large (10^5 elements) and not all elements/properties are displayed on diagrams.
- Gaps/inconsistencies can be difficult to detect visually.
- Grading must be completed promptly (since most deliverables are additive on prior work).
- “Bad” models take more time to grade since thoughtful comments and suggestions are needed to help students improve their ongoing model’s quality.

Automated Validation

Lessons Learned and Recommended Best Practices from Model-Based Systems Engineering (MBSE) Pilot Projects, Ryan Noguchi, Aerospace Corporation (2016)

“

…many syntactical errors that would have been caught had they used the built-in model validation capabilities of their tools, but the problem would also have been apparent upon visual inspection by an experienced modeler… it is critical that model reviews be performed frequently by experienced modelers, particularly to check for semantic mistakes—those that won’t be caught by the modeling tools’ validation checks…

Model reviews performed in a briefing format or through static captures of the model (typically via PDF files or HTML files) are much less effective at ferreting out errors.”

Accelerating Training via Validation

Shipshape and Bristol Fashion: Model documentation and curation to facilitate reuse

Michael J. Vinarcik, Chief Solutions Architect & Digital Engineering Strategist

Heidi Jugovic, Chief Solutions Architect & Digital Engineering Strategist

2019 NDIA Mission and Systems Engineering Conference

- A strong validation suite acts as a “digital mentor” to new and improving modelers… a mentor who consumes ***no ongoing labor hours***
- If the validation suite is structured correctly, it provides feedback that follows educational guidelines for effectiveness
  - **Goal-oriented**: Reduce errors to zero!
  - **Prioritized**: Severity tells you what to work first
  - **Actionable**: A well-written error message tells the modeler exactly how to fix the error
  - **Learner-friendly**: No embarrassment of being constantly corrected by a colleague
  - **Ongoing, Consistent and Timely**:
    - The rules will be enforced the same, every time
    - Everyone gets the same feedback
    - Anyone gets feedback on demand, before too many errors have accumulated

Classroom Benefits of Automated Validation

- Students can focus on the intellectual content of their models because basic syntax, completeness, and consistency checks are provided on demand.
- Instructors benefit from faster grading since validation detects fundamental errors and style violations.
  - Skim diagrams for obvious issues
  - Focus on any areas with existing errors (likely indicative of more pervasive issues)

SAIC Digital Engineering Validation Tool Evolution

- More than 4,000 downloads since its initial release
- Provided for free as a service to the worldwide modeling community at https://www.saic.com/digital-engineering-validation-tool

VERSION	DATE	# OF RULES	HIGHLIGHTS
V1.0	Dec 2019	126 rules	• Initial customizations • Videos
V1.5	Apr 2020	153 rules	• Model-based Style Guide • Example model (Ranger lunar probe) • Rhapsody rules
V1.6	Aug 2020	168 rules	• Classification/Data Rights customization
V1.7	Jan 2021	184 rules	• FMEA customization
V1.8	Jul 2021	192 rules	• UPDM rules (beta)
V1.85	Oct 2021	194 rules	
V1.90	Feb 2022	201 rules	
V2.0	Aug 2022	220 rules	• Includes model federation process and rules
V2.5	Jun 2023	226 rules	• Added UAF rules and native 2021x support
V2.6	Dec 2023	236 rules	• Added SW4SysML optional profile
V2.7	Jul 2024		• Revised port input/output rules

Published works related to the growth and use of the SAIC Digital Engineering Validation Tool validation rules

- ***Treadstone: A Process for Improving Modeling Prowess Using Validation Rules***
  2020 American Society for Engineering Education Annual Conference and Exposition
- ***Using SysML State Machines to Automatically Conduct Failure Modes and Effects Analysis***
  2020 NDIA Systems & Mission Engineering Conference
- ***Inconceivable: Those Requirements Don't Mean What You Think They Mean***
  2020 NDIA Systems & Mission Engineering Conference
- ***Treadstone + 1: The First Anniversary of the SAIC Digital Engineering Validation Tool***
  2021 INCOSE International Workshop MBSE Lightning Round
- ***A State-Based Approach for ESOH Analysis***
  2021 NDIA Systems and Mission Engineering Conference
- ***Outcome: Rules-Based Training and Development for System Modelers***
  2021 INCOSE Great Lakes Regional Conference
- ***A Mars Octet: Lessons Learned from Federating Eight Student Models in a SysML Class***
  2022 AIAA SciTech Forum and Exposition
- ***Good Fences Make Good Neighbors Principles for Model Federation***
  2022 NDIA Systems and Mission Engineering Conference
- ***Here There Be Dragons: An Initial Study of Undetected Errors in Unvalidated SysML Models***
  2023 MBSE Cyber Experience Symposium
- ***True or False: How to Craft Automated Validation Rules***,
  2024 MBSE Cyber Systems Symposium
- ***Automating Rule-Checking to Identify SysML Modeling Errors: A Preliminary Study in a Classroom Environment***
  2024 INCOSE International Symposium

Initial Assessments of Validation Impact

Project Evolution

Phase I
File-based document

Year	Course	Topic
2012	AEV 5060	Ultra Probe*
2012	AEV 5070	Battery Electric Vehicle
2013	AEV 5060	Rescue and Exploration EV
2013	MPD 5050	ReallyLongRiver Delivery System
2014	MPD 5100	Henry Ford Museum Systems
2015	MPD 5050	Portable Sustainment Pod

Phase II
Collaborative, single projects

Year	Course	Topic
2016	MPD 5100	PRZ-1 Notional Nuclear Submarine*
2016	MPD 5050	NeMO*
2017	MPD 5100	NeMO (Continued)
2017	MPD 5050	Portable Sustainment Pod
2018	MPD 5100	FIRST Robotics
2018	MENG 5925	NeMO Hypermodel

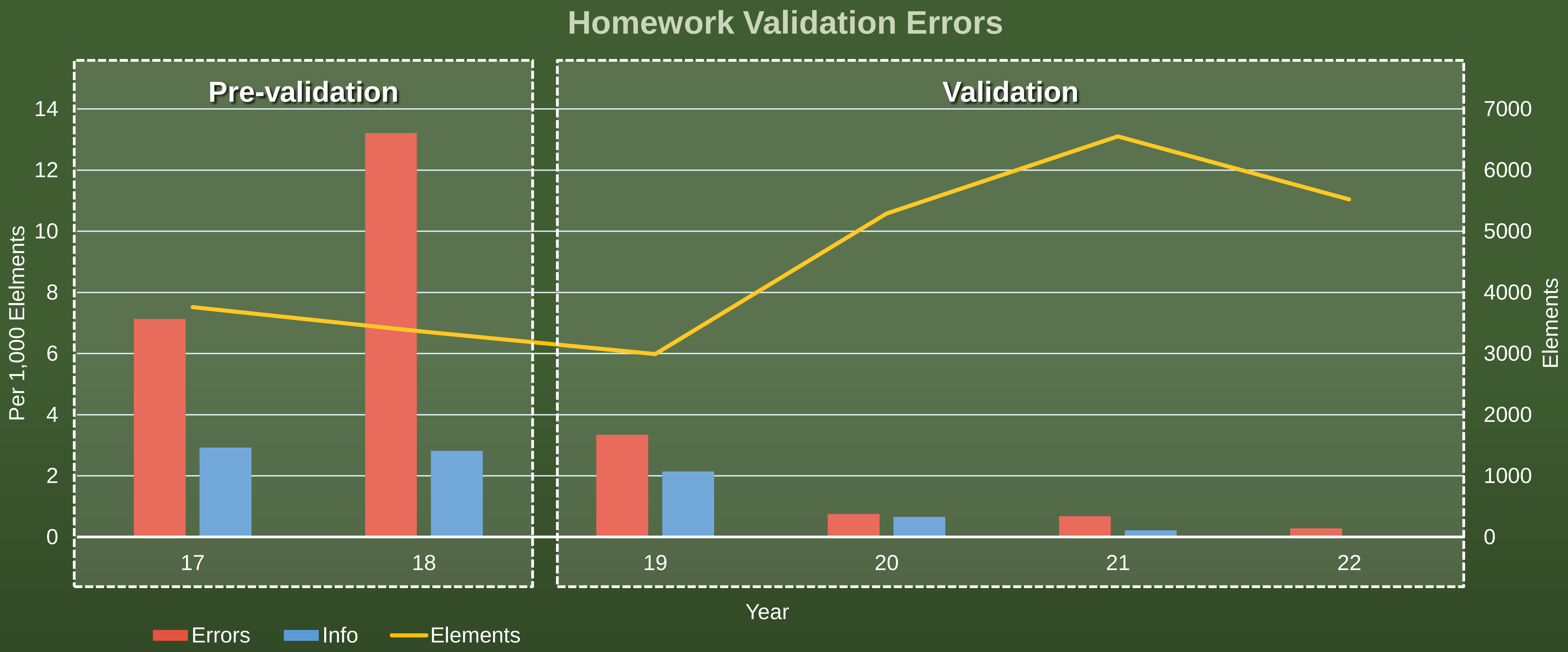
Phase III
Federated / Validated

Year	Course	Topic
2019	MENG 5925	Mars Rovers*
2020	MENG 5925	Project Tin Hyper model Cubesat
2020	MENG 5925	Mars Octet*
2021	MENG 5925	Space: 1999
2022	MENG 5925	Lunar Architecture

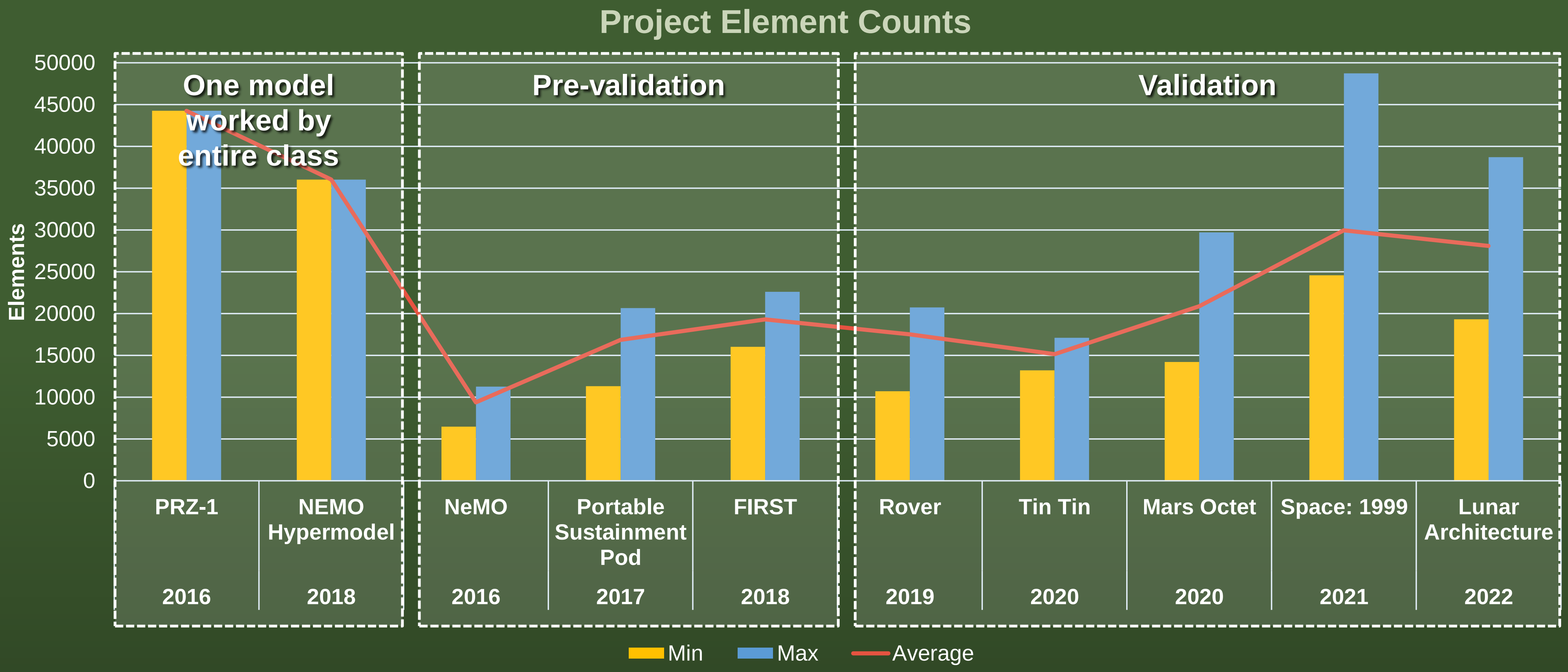
* Indicates a publication detailing the project is available on academia.edu

From *Forged in Fire: Teaching the Craft of Model-Based Systems Engineering*, Vinarcik, 2023 INCOSE International Symposium

MENG 5925 Homework Errors per 1,000 Elements



MENG 5925 Project Size



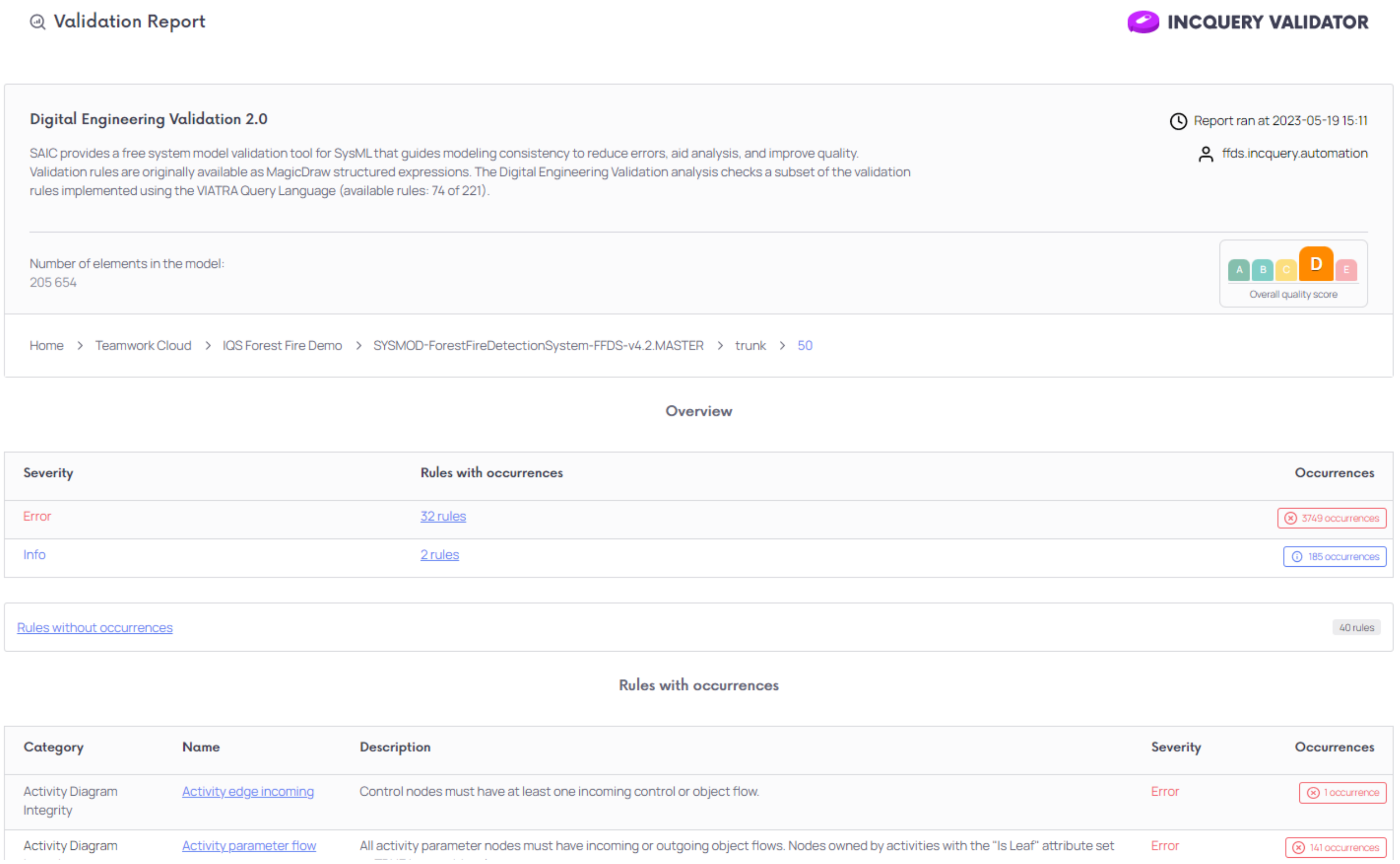
Analysis

Methodology

- Cameo System Modeler/MagicDraw and a TeamWork Cloud repository are used to construct the student models.
- Only models related to term projects were included in this study; tool-provided profiles were excluded from the analysis.
- IncQuery Labs (www.incquery.io) applied their Validator tool to the models on the server, capturing errors and relevant metadata to facilitate this study.

IncQuery Validator

https://incquery.io/validator



Validation Report

INCQUERY VALIDATOR

Digital Engineering Validation 2.0

Report ran at 2023-05-19 15:11

ffds.incquery.automation

SAIC provides a free system model validation tool for SysML that guides modeling consistency to reduce errors, aid analysis, and improve quality. Validation rules are originally available as MagicDraw structured expressions. The Digital Engineering Validation analysis checks a subset of the validation rules implemented using the VIATRA Query Language (available rules: 74 of 221).

Number of elements in the model:
205 654

A B C D E
Overall quality score

Home > Teamwork Cloud > IQS Forest Fire Demo > SYSMOD-ForestFireDetectionSystem-FFDS-v4.2.MASTER > trunk > 50

Overview

Severity	Rules with occurrences	Occurrences
Error	32 rules	3749 occurrences
Info	2 rules	185 occurrences

Rules without occurrences — 40 rules

Rules with occurrences

Category	Name	Description	Severity	Occurrences
Activity Diagram Integrity	Activity edge incoming	Control nodes must have at least one incoming control or object flow.	Error	1 occurrence
Activity Diagram	Activity parameter flow	All activity parameter nodes must have incoming or outgoing object flows. Nodes owned by activities with the "Is Leaf" attribute set	Error	141 occurrences

Fields in Dataset

Field	Definition
Model	Model name
Group Path	TeamWork Cloud Category Name
Category	Rule Category (e.g, Completeness, Structural Integrity)
Rule name	Rule name
Symbolic name	Same as Rule Name
Severity	Error, info, or warning
Message	The rule's error message (typically explains what it is and how to correct it)
Element ID	The Cameo native element ID for the violating element
Element name	The Cameo native name for the violating element
Element type	The type/metaclass for the violating element
Element link	A Cameo-native URL
Scope size	Not used at this time
Model root name	The name of the root model element
Model root type	Primary project or Used project

Analysis Definitions

- Errors were considered violations of a given rule.
- Error situations were also defined.
  - **Fundamentally unaware**: Rule does not exist.
  - **Unaware**: Rule implementation has changed.
  - **Cognizant**: Rule is under development.
  - **Aware**: Rule is present in the ruleset.
- Fundamental violations are errors or omissions related to SysML.
- Stylistic violations are related to stylistic conventions and requirements.

Error Analysis

- 1.474M errors were logged by the tool; a reduced set of fifty-three project models (from 2016-2022) was selected for analysis; these were associated with 107,644 errors.
- The collected data were statistically analyzed by combination of SQL databases and Python dataframe structures; this reduced the number of errors under study to 23,223.
- To detect rule evolution, word and sentence similarity measures from Python's Spacy library were used; these were inaccurate so a direct alphanumeric sequential comparison was used.

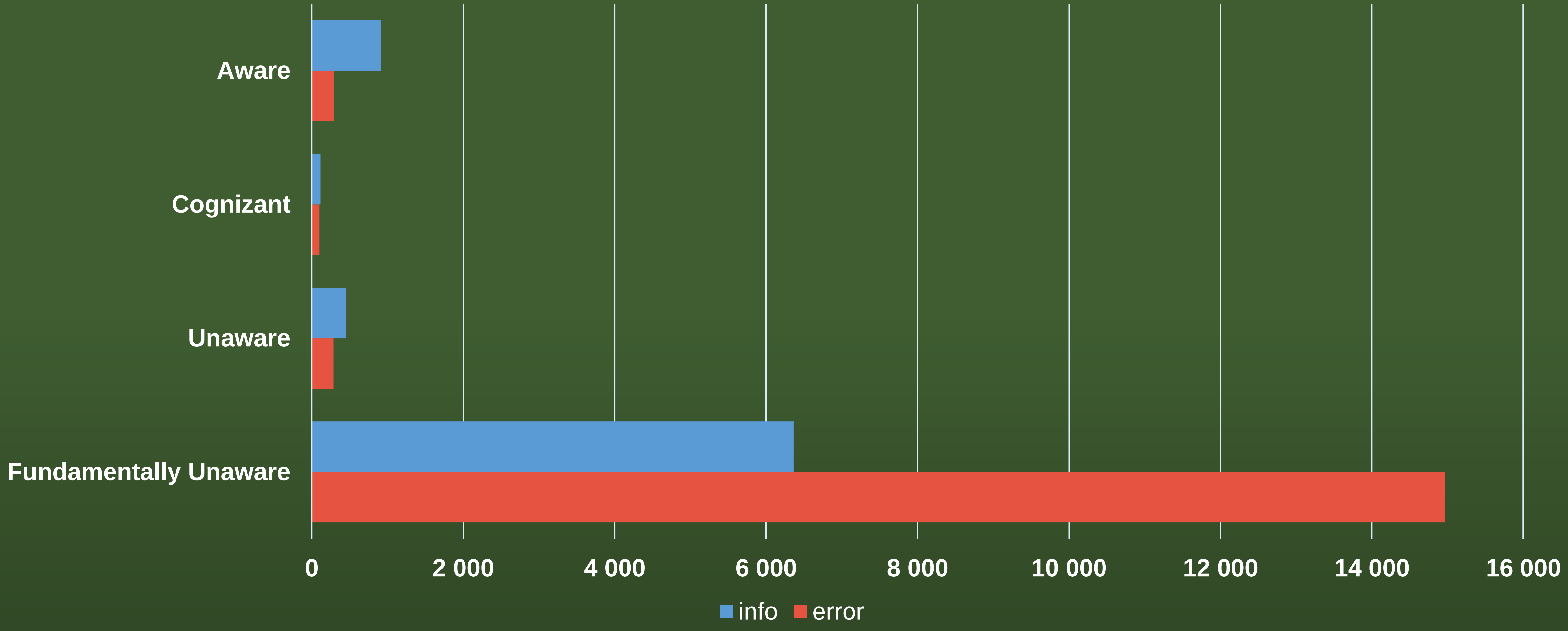
Error Counts (situation vs. severity)

Situation	Error	Info	Warning
Unaware	14,982	6,380	333
Cognizant	324	275	0
Aware	507	1,349	2

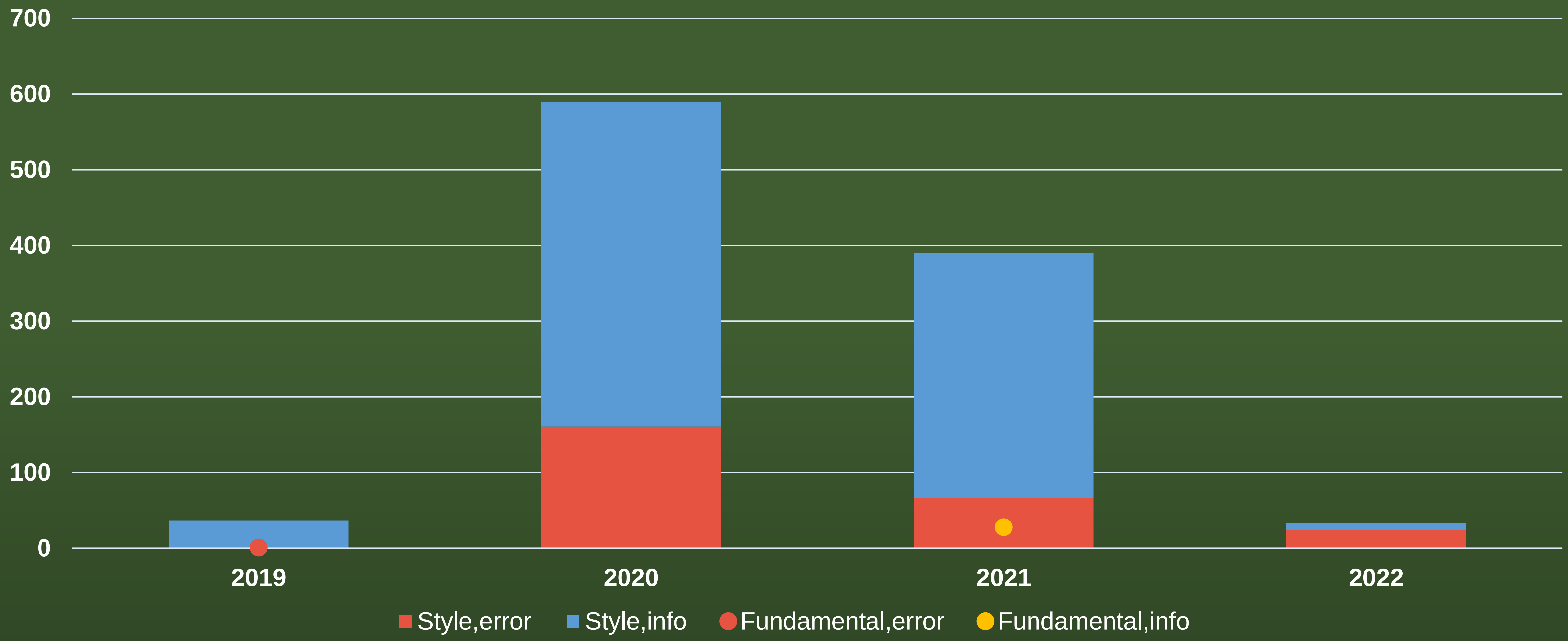
Results

- 97% of violations are from unaware students and 90% are present in models that lacked any automated validation.
- Outlier models were removed (These were impacted by late interface changes and had a large number of item flows; these students were given special dispensation and were not expected to submit models with zero errors.)

Violation Counts vs. Situation



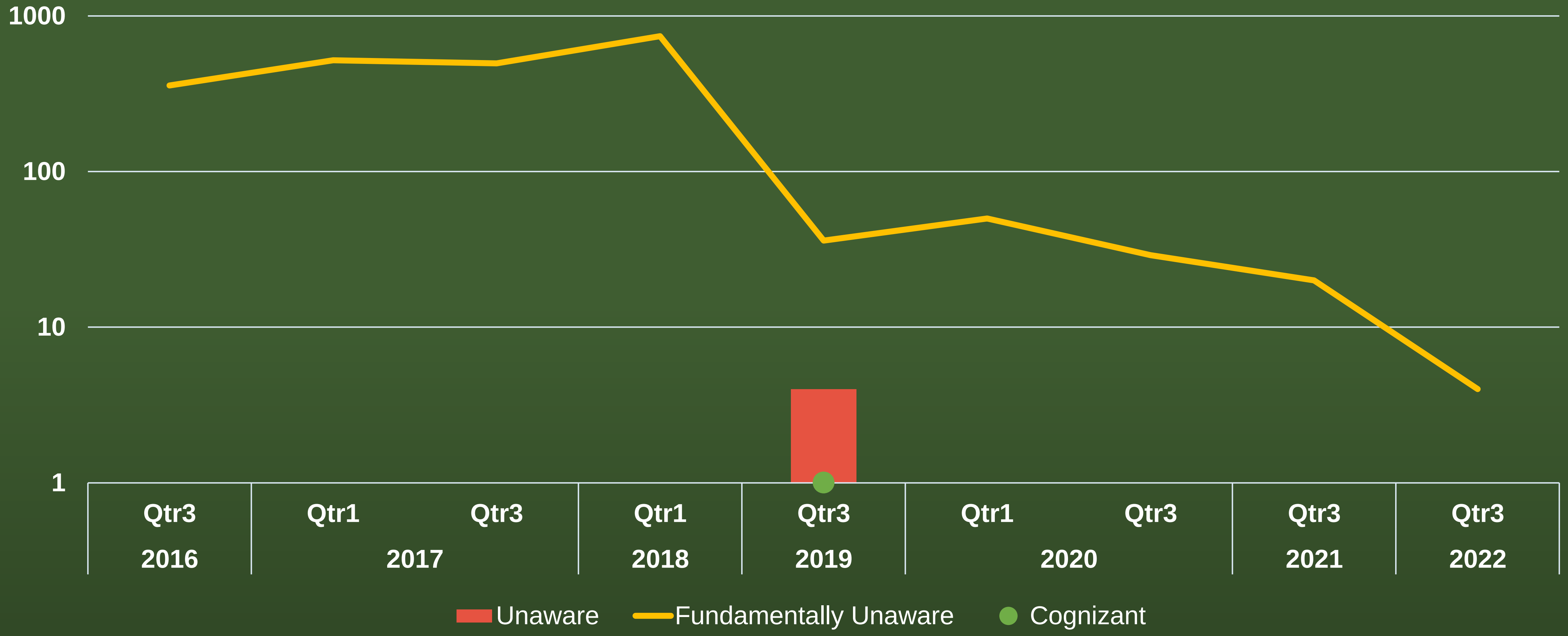
Fundamental and Style Violations Over Time



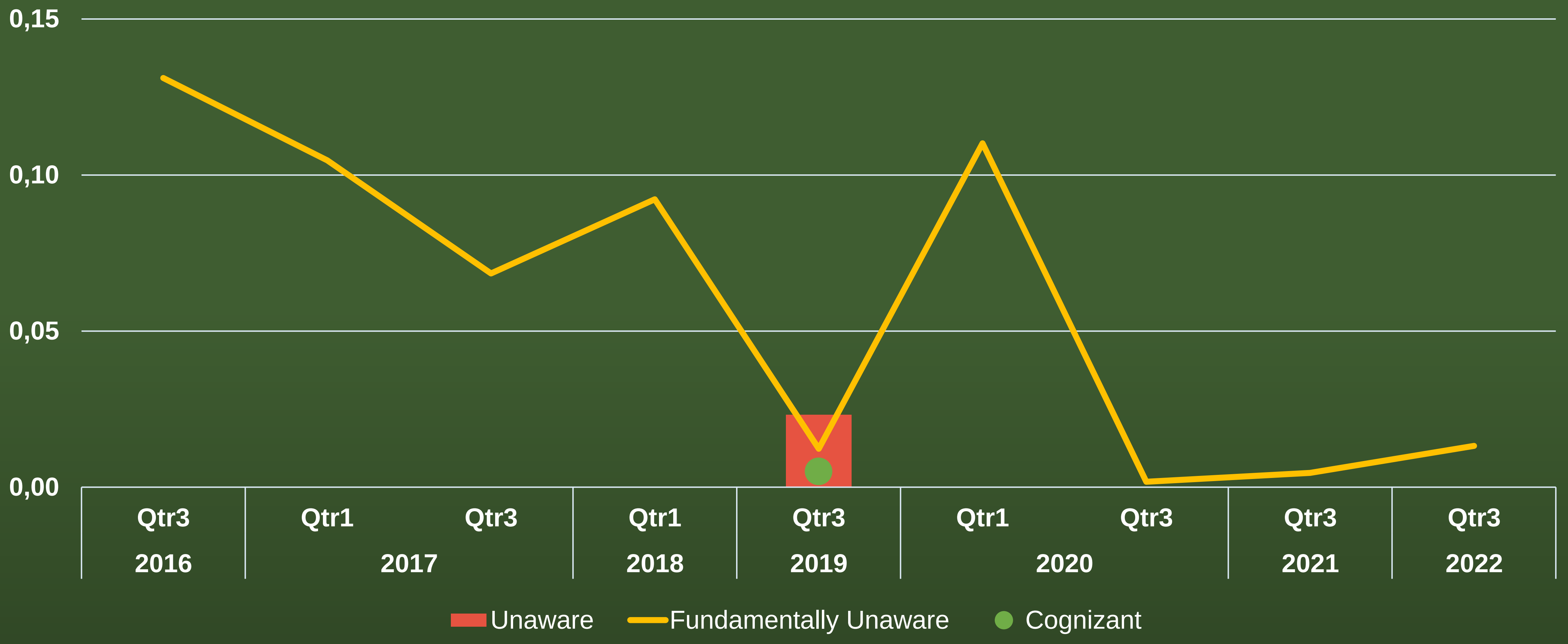
Timeline Analysis

- Timeline graphs were built to visualize how the violations have evolved over time under different situations.
- There are minimal latent violations, and none were found to be made in aware situation.
- Per-model error ratio, which is calculated by normalizing violation counts by the models' element counts, show that non-unaware violations showed not only lower frequency but lower violation ratio as well.

Fundamental Error Counts Over Time



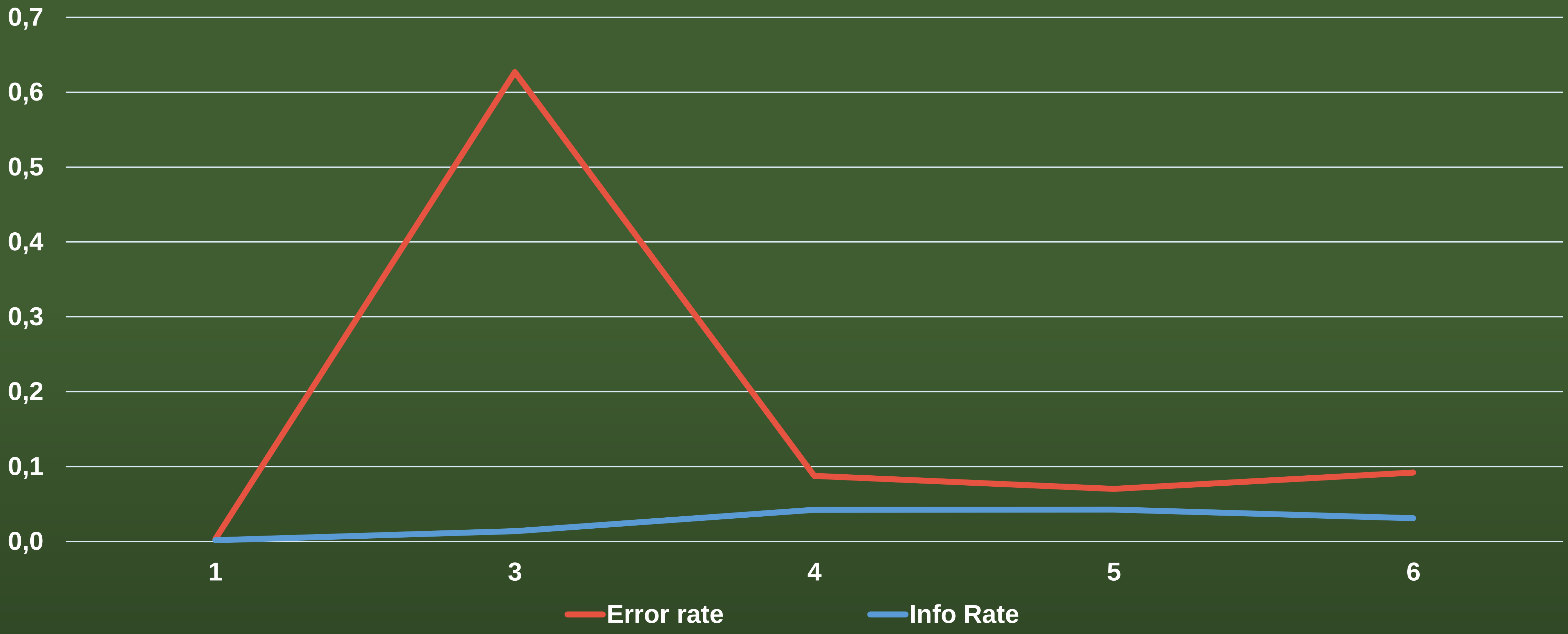
Fundamental Error Rates Over Time



Violation Rates as a Function of Group Size

- Single student models had the lowest violation rates (Note: Only one model was constructed by a single student.)
- 4-6 students per model had lower violation rates than 3 students.

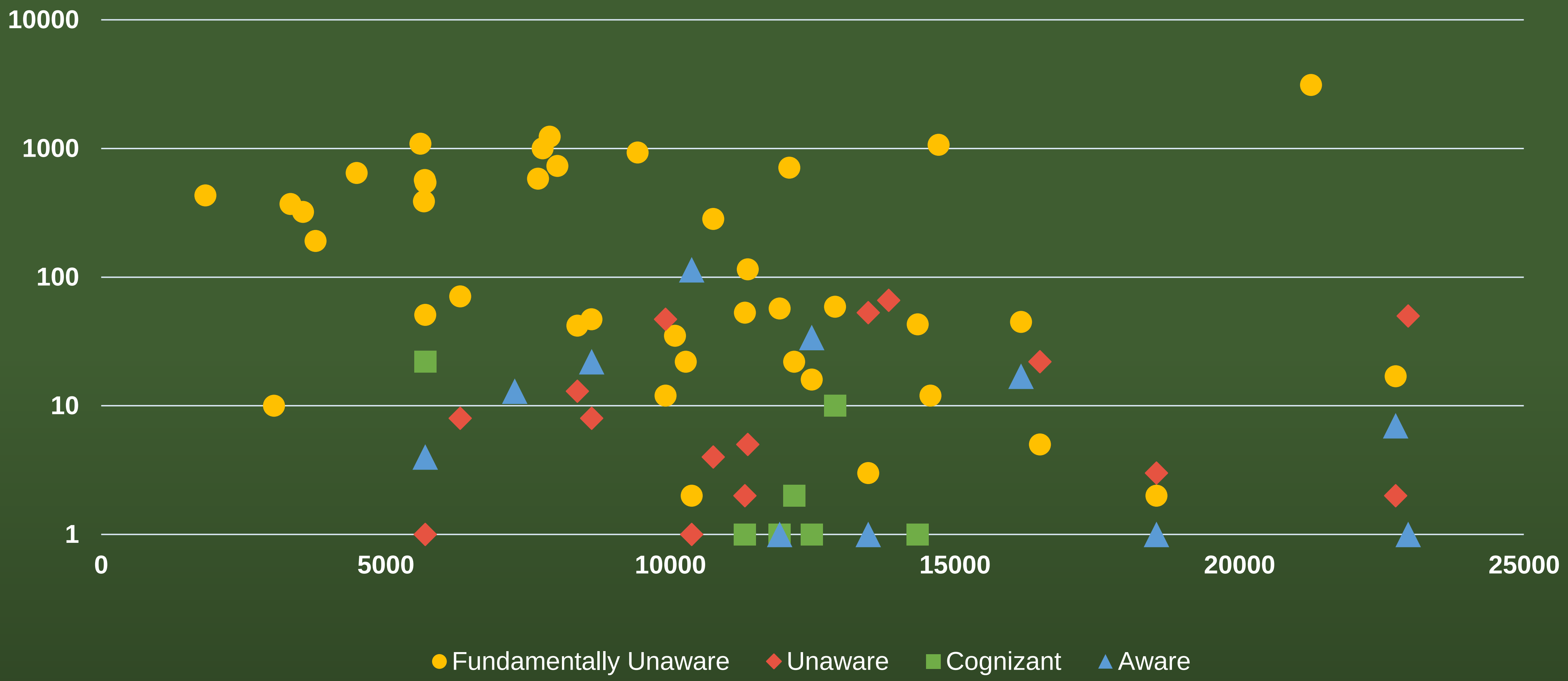
Error / Info Rates as a Function of Group Size



Analysis of Violation Counts vs. Element Counts

- Fundamental and stylistic violations were plotted.
- No clear trendline was detected.
- Polynomial regression with degree 1 to 3 as well as logistic regression was tested with the per model element count as an independent variable.

Violation Counts vs. Model Element Counts



Conclusions

- Categorization by violation situations and style of the rules showed that the majority of errors were made by unaware students and/or for stylistic rules.
- Per-model error ratio was used to show that while there are slight differences, the overall frequency pattern follows the relative ratio per models.
- The study showed that the term project models resulted in nearly zero latent errors when non-stylistic rules are concerned, with most of the latent errors categorized as stylistic rather than fundamental violations.

Future Work

- Future work related to this dataset may include additional analyses of the error rates over time or a comparative study, in which a project is replicated without the use of MBSE or SysML.
- This would facilitate a comparative analysis to illustrate the impact system modeling has on traceability, rigor, quality, and completeness.

THANK YOU!

Michael J. Vinarcik, ESEP-Acq, FESD
vinarcmi@udmercy.edu

Sukhwan Jung, Ph.D.
shjung@arizona.edu

Alejandro Salado, Ph.D.
alejandrosalado@arizona.edu

34th Annual **INCOSE**
international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

www.incose.org/symp2024
#INCOSEIS