34th Annual INCOSE international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

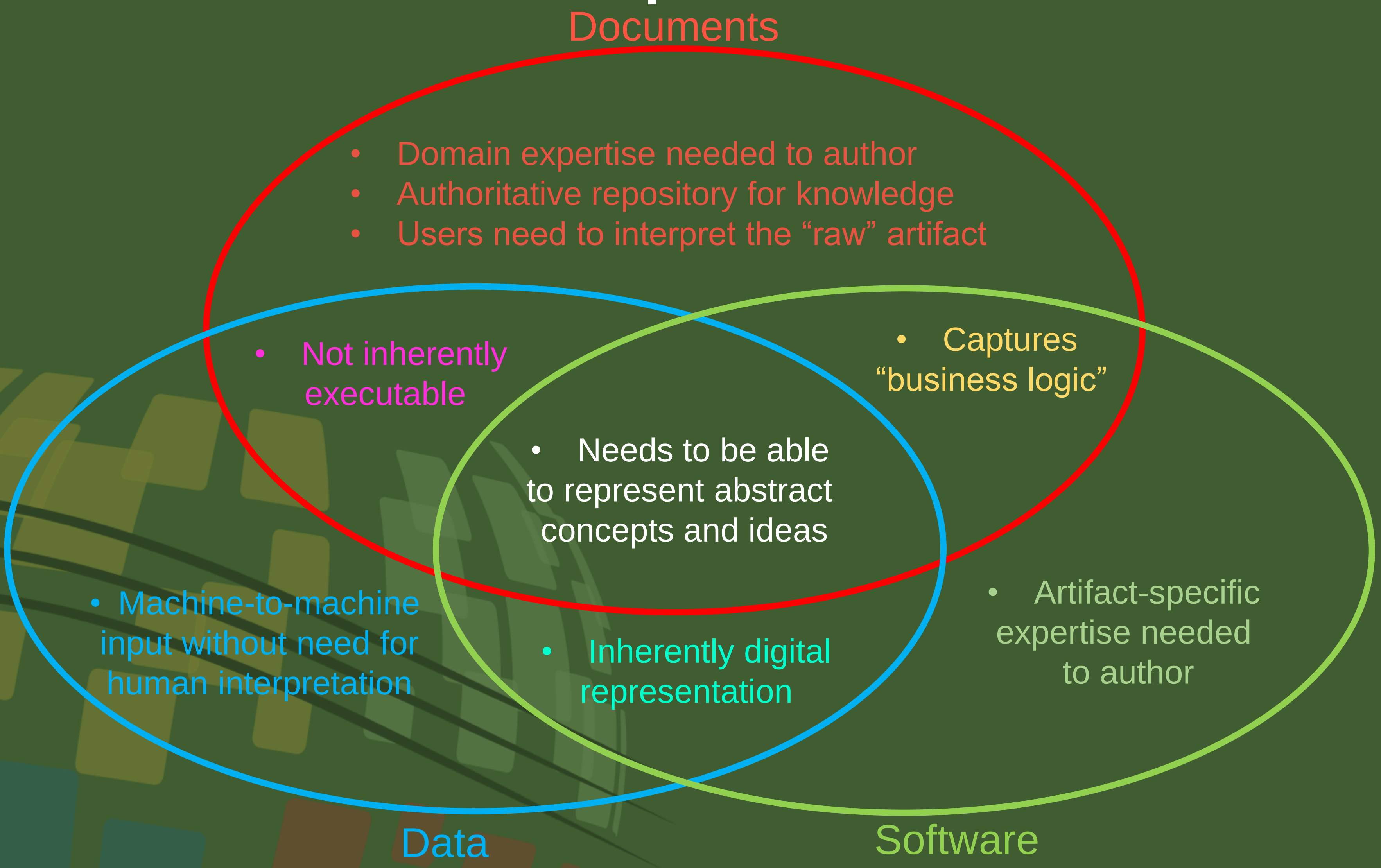
The Nature of Technical Debt in the Development of Descriptive Models in MBSE

Ryan Noguchi, The Aerospace Corporation

The Nature of MBSE

MBSE is the emerging practice of SE
in which **descriptive models** are the
authoritative embodiment of SE knowledge

The Nature of Descriptive Models



Technical Debt—What is it?

A metaphor for development or sustainment costs deferred to the future

Repair, rework, replace, refactor, re-engineer, ...

created “when developers violate good architectural or coding practices, creating structural flaws in the code.” – Curtis, et al.

Like financial debt, it can be a useful tool ...
when used at the right time for the right purposes

Principal and Interest

Principal:
The cost to avoid incurring debt

Take the time to “do it right” now

Interest:
The additional cost to “make it right” later

Interest costs typically increase with time

Taxes!

Taxes:
Additional costs on <u>transactions</u>
resulting from the suboptimal implementation

Taxes can impact different stakeholders differently

Taxes result in <u>lost productivity</u> and <u>lost value</u>

Death!

The model dies when it isn't worth fixing

The interest payment is too high

Taxes are too high

Often easier to just start over

This can be okay ... if the model is intended to be temporary

The Calculus of Technical Debt

The goal is not to eliminate debt …
but to make better informed choices about debt

Technical debt is not deterministic
What constitutes debt changes as the context changes

In Agile software, risk of rework is less than risk from delayed feedback
This makes technical debt more likely to be worthwhile

The Calculus of Technical Debt for Models

Models are different!

Model rework is often much more difficult than for software

More often architectural in scope and nature

Implementation details are exposed to users

Refactoring of models is less doable piecemeal

Automated discovery is more difficult

Defect “contagion” is common

Technical debt is riskier for models than for software

Characterising Technical Debt

Scale/Growth	How many elements? How common are they?
Propagation	How quickly and how much do impacts spread?
Rework	How difficult is the rework?
Use Taxes	What are the impacts on model builders and users?

Modelling for Purpose

Biggest obstacle for success is lack of:

Well understood,
explicitly defined and documented,
sufficiently compelling
purposes for the models to fulfill

Without knowing these purposes, you're modelling in the dark!

Without knowing these purposes, you can't make well-informed decisions about how to architect and implement the model

Technical Debt in Model Implementation

Model implementation is exposed to model users

Unlike software source code

Model technical debt is mostly driven by model architecture

But often arises at the implementation level

What are some principles for moderating implementation debt?

Key Model Implementation Principles

Don’t Repeat Yourself	Minimize redundant or duplicative information
Avoid Brittle Views	Build views to be readily sustained as the model grows
Cite Your Sources	Proactively document information sources
Use Units	Avoid unitless value properties
Build Quality In	Build quality in—don’t try to retrofit it later

Summary

Technical debt is rework deferred to the future for expediency

Understand how technical debt influences model value

Understand the technical debt implications of
model architectural and implementation decisions

Make model architectural and implementation decisions
to avoid unnecessary technical debt

34th Annual INCOSE international symposium

hybrid event

Dublin, Ireland
July 2 - 6, 2024

www.incose.org/symp2024
#INCOSEIS