Technical Debt Mitigation in Requirements using Artificial Intelligence

Krish Patel, 2024 INCOSE NEFWS

Draper Scholar Program

Agenda

- Research Overview & Background Information
- Choosing an Artificial Intelligence (AI) Environment
- Future Methods for Data Pipelining

Research Overview and Background

Understanding Technical Debt (TD) within Systems Engineering (SE)

Research Overview

Technical Debt (TD) has moved beyond software engineering into full engineering lifecycles

Minimal existing documentation classifying and mitigating technical debt in Systems Engineering (SE) Lifecycles – specifically in Requirements planning

Create AI powered solution to identify and mitigate TD in Draper SE projects

Project Overview

- 2-year project, 3 phases

Classification

- Duration: 6 months
- Select SEMP at Draper/WPI
- Develop a TD/LI classification table/map
- Develop training data set

Training

- Duration: 1 year
- Select LLM
- Set up AI architecture
- Prepare and train the dataset

Testing

- Duration: 6 months
- Test the model on another project to identify TD
- Build TD tracking over the project lifestyle
- Build prediction of risk-based TD

Technical Debt

The gap between making something perfect and making it work

Technical debt is a practice of trading short term gains for long term benefits

- Allows companies to rush products to market and generate revenue sooner
- Often gets paid back, with interest

Why take on technical debt?

- The cost of fixing something later is cheaper than the cash investment to take a complete product to market

Types of Technical Debt

Design/Code TD	• Amassed debt from poor design and planning, lack of consideration for certain features
Requirements TD	• Difference between the ideal value of the specification and the actual implementation of a system.
Test TD	• Suboptimal testing procedures, or lack of testing for certain features
Build TD	• Suboptimal build procedures for code pipeline
Documentation TD	• Missing or inadequate documentation of any type
Infrastructure TD	• Delayed upgrade decision
Defect TD	• Known defects that aren't fixed

Connecting Technical Debt to Systems Engineering

Understanding Leading Indicators in Systems Engineering

- Measure for evaluating the effectiveness of how a specific activity is applied on a project in a manner that provides information about impacts that are likely to affect system performance objectives
- Could be a single metric or collection of measures
- Generally predictive of future systems engineering project performance
- Aid leadership in delivering value to customers and end users

Connecting Technical Debt to Systems Engineering

- Leading Indicator Trend Examples

Schedule & Cost Pressure

- Impact of schedule and cost challenges on carrying out a project

Process & Compliance Trends

- Availability of non-personnel resources (infrastructure, capital assets, etc.) needed throughout the project lifecycle.

Requirements Verifications Trends

- Progress against plan in assuring that the customer requirements are valid and properly understood.
- Adverse trends would pose impacts to system design activity with corresponding impacts to technical, cost & schedule baselines and customer satisfaction.

Technology maturity trends

- Risk associated with incorporation of new technology or failure to refresh dated technology.
- Adoption of immature technology could introduce significant risk during development while failure to refresh dates technology could have operational effectiveness/customer satisfaction impact

Impact of Technical Debt on System Dimensions

Baseline System Design → System takes deliberate debt → Debt has grown out of control → Debt is handled at the trade-off of speed and profitability

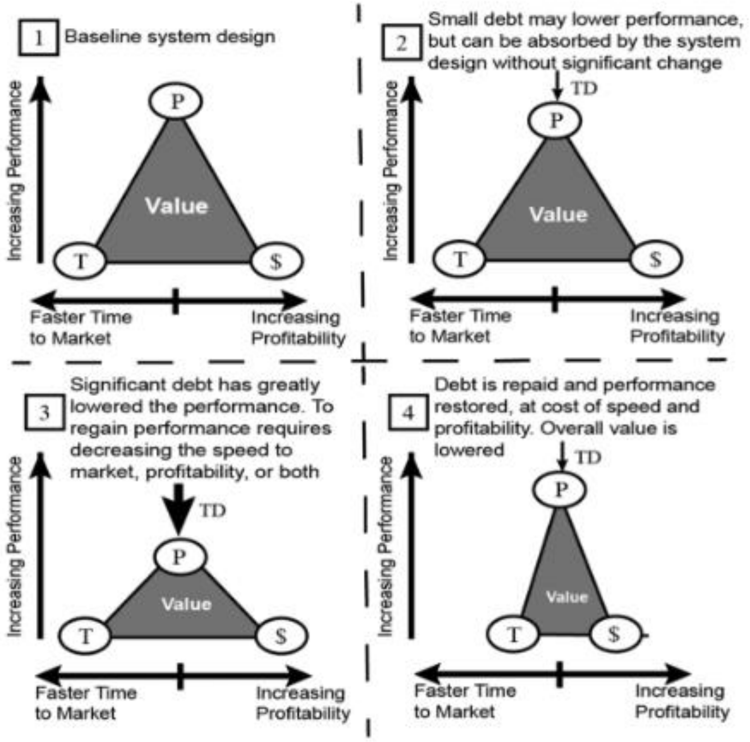


Figure 1. Impact of TD on project schedule, performance, and cost during project execution (Klienwaks et. al, 2023)

Connecting Technical Debt to Systems Engineering

- TD to LI Mapping Case Study; Professor Shamsnaz Bhada

Leading Indicator Trends and Technical Debt are linked in Systems Engineering:

- **Both are used to predict and prevent potential problems in a system**
- Analyzing LI's lets engineers identify potential problems and take measures to prevent them
- Analyzing TD allows engineers to reduce the potential for future problems and improve efficiency of the system

Connecting Technical Debt to Systems Engineering

- TD to LI Mapping: Hubble Telescope Failure Documentation Case Study

Event: “P-E and NASA (National Aeronautics and Space Administration) **both understood and accepted this approach despite a lack of independent measurements to confirm the reliability of the primary test**. The failure was not one of system engineering design, **but rather one of manufacturing system design and process/quality control**. This event occurred at a time when there was also great **concern about cost and schedule, overshadowing the obvious need for independent verification testing**, or attention to the anomalous RNC data suggesting that something might have been wrong.”

TD Type: Test Debt

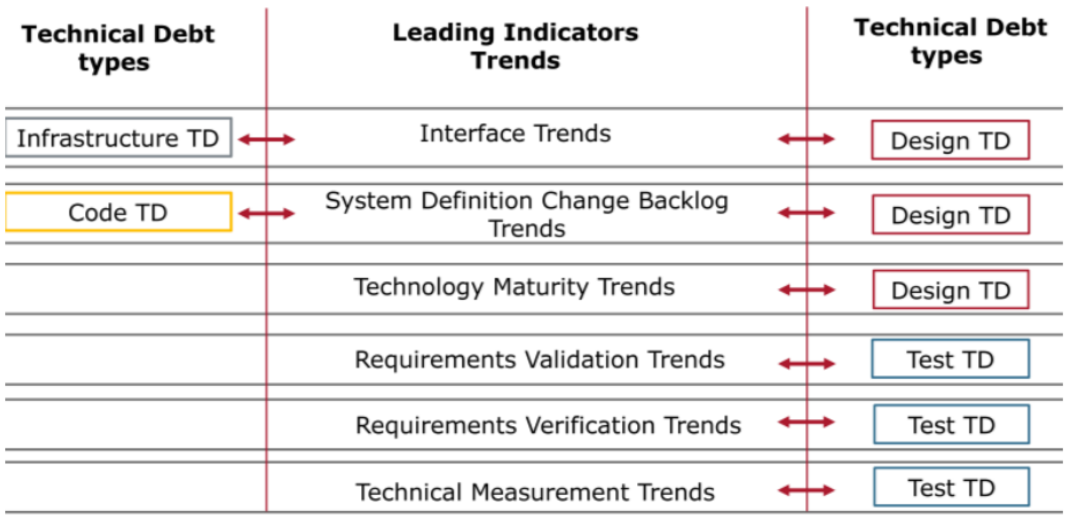
Leading Indicator: Schedule and Cost Pressure

Connecting Technical Debt to Systems Engineering

- Overall Mapping Patterns from Case Study

- Risk exposure and handling trends were omitted due to already being documented in risk registers

TD Type	Leading Indicator
Test Debt	Schedule & Cost Pressure
Documentation TD Test TD	Process Compliance Trends Requirements Verification Trends
Design TD Architecture TD Test TD	Technology Maturity Trends



Choosing an Artificial Intelligence Environment

Solution use cases

- Example Prompts for the LLM

Help me identify all instances of TDs & LI's in the build/eval stage of my project. Here are the relevant documents (SEMP, project lifecycle plan, etc.)

Here are some LI's I think I've noticed in my project. What technical debts should I be aware of, and how can I fix this?

Here is a partially completed SEMP and lifecycle document. Could you finish the project in a manner that limits and current and future technical debt?

Choosing an Artificial Intelligence Environment

Understanding Intelligent Document Processing (IDP)

Use of AI technologies to automate extraction, processing, and management of input documents

Core Components:

Provides increased scalability, security, and availability over manual document intake

- Optical Character Recognition (OCR)
- Natural Language Processing (NLP)
- Robotic Process Automation (RPA)

Choosing an Artificial Intelligence Environment

Natural Language Processing for Requirements Engineering

NLP: subset of AI that enables computers to understand, interpret, and generate human language

NLP4RE: subset of NLP that specifically focuses on requirements engineering

- Automates requirements analysis
- Improves quality of requirements
- Classifies and validates requirements to maintain compliance with preset rules

Choosing an Artificial Intelligence Environment

- Further analysis of IDP solutions: What makes a good environment for AI growth?

Security	• AI and Document Environment must be capable of public/private network configuration, and access control at the resource level
Scalability	• The Environment of choice must be able to scale resources horizontally and vertically to adapt to sudden bursts.
Performance	• IDP Solution must be able to characterize and classify text to a high degree, dealing with both computer-readable documents and images
Automation	• The Environment must have the adaptability to be run at the script level, offering the ability to create triggers on document upload

Choosing an Artificial Intelligence Environment

- IDP Comparison Chart: Cloud Offerings are supreme to locally developed IDP solutions

Environment Indicator	Amazon Web Services	Microsoft Azure	DocumentAI by Google	Manual via Google Colab
Security	9	9	7	4
Scalability	10	10	7	5
Performance	9	9	7	5
Automation	9	9	6	4

Choosing an Artificial Intelligence Environment

Retrieval Augmented Generation: Additional Context for LLM prompting

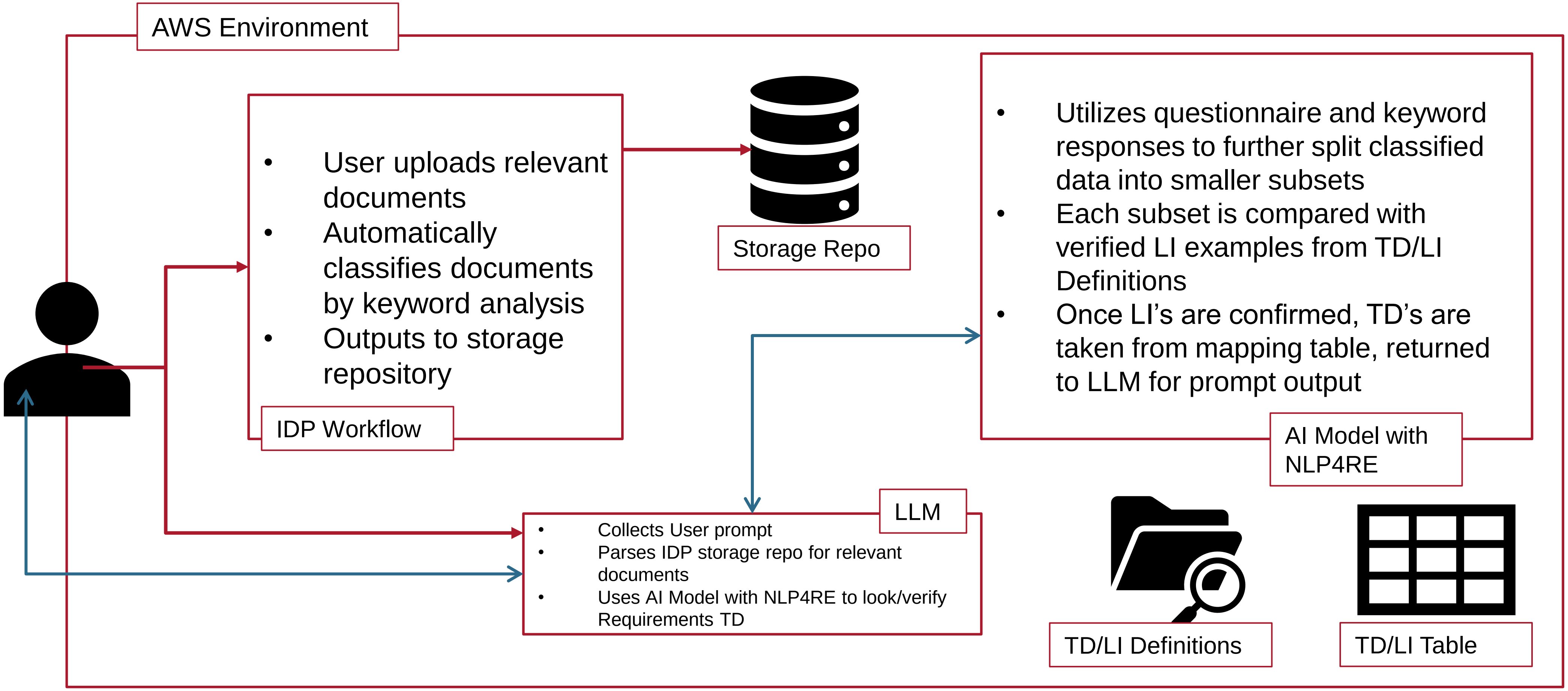
Many LLM prompts are often document heavy and require a large amount of processing

Most LLM's are limited in the amount of the contextual data they can pull for each prompt, creating a gulf in data processing for document heavy applications

Retrieval Augmented Generation (RAG) Method:

- Takes documents of similar relevance, feeds it back into the LLM as input
- Layered prompting, allows users to get more verification with LLM answers

Proposed Architecture



DRAPER® SCHOLARS WPI

Next Steps

Future methods for data pipelining

Requirements TD Analysis Data Pipeline

Surveying SE Professionals to understand and digitize Requirements TD Evaluation

Survey and Analysis:

- Conduct surveys with SE professionals about their knowledge and technical background and provide them with passages from requirements profiles to highlight requirements debt, recording and evaluating their process.

Documentation and Conversion:

- Convert the recorded processes into written documents, then into a computer-readable format, and feed this structured knowledge into a large language model.

Model Applicaiton & Validation:

- Apply the data to ML models, validate their effectiveness, and train the models on the SE professionals' thought processes and language.

Strategy Determination:

- Determine the optimal strategy by evaluating in-context learning and chain-of-thought reasoning, and implement a hybrid approach if necessary, iterating and improving the models.

Using LLMs with a thought process

In-context learning vs. Chain of thought

In-context learning (ICL):

- Learns by example within the prompt input, adapting based on the patterns or tasks provided in real-time without updates to the underlying model
- Focuses on pattern recognition from provided examples, requiring the user to supply relevant context and demonstrations to guide the model's behavior

Chain-of-thought (CoT):

- Explicitly models reasoning steps, breaking down complex tasks into sequential, logical steps to reach a solution.
- Encourages step-by-step reasoning and deliberation by modeling intermediate steps to reach an answer, often yielding improved performance in complex tasks

Data Pipeline with LLM involvement

Potential solution usage scenario upon completion of analysis pipeline

Document Ingestion & Intitial Analysis

- User provides a new document, which is fed into the LLM for initial NLP4RE analysis to extract preliminary insights.

Thought Process Integration:

- The LLM utilizes pre-existing thought processes to refine the initial analysis, applying deeper contextual understanding.

User Interaction & Prompting:

- Users can prompt the LLM for further analysis, enabling real-time, dynamic interaction and additional insight extraction.

Feedback & Continuous Refinement

- The analysis is continuously refined based on user prompts and feedback, updating the document with new, iteratively improved insights.

Model Training & Deployment

- Updated insights are integrated into ML models, which are validated, trained, and then deployed for ongoing requirements debt detection and mitigation, adapting with new data and user interactions.

Thank You for Listening!

Questions?