Development of a Software Platform for Pipeline Health Monitoring and Management

Mihai A. Diaconeasa* **and Ali Mosleh**
The B. John Garrick Institute for the Risk Sciences
Department of Materials Science and Engineering
University of California, Los Angeles, USA

Abstract: This paper describes the software requirements and architecture of a risk-based pipeline integrity management support tool to aid in decision-making and planning by the pipeline operators. This platform is being developed as the main product of the research project on Pipeline System Integrity Management sponsored by the Petroleum Institute, Abu Dhabi, UAE, in collaboration with a large interdisciplinary team from the sponsoring agency and the University of Maryland, Department of Mechanical Engineering. As such, the platform design is supported by a multi-disciplinary science and engineering approach for a comprehensive, state-of-the-art solution. Where possible, existing technologies and methods are leveraged, and new ones are developed as needed to meet the objectives.

Keywords: Pipeline Integrity.

1. INTRODUCTION

Transportation of oil and gas products through pipelines is one of the most reliable, safe, and cost competitive ways of delivering large volumes of products over long distances. Nevertheless, the pipeline design, operation, and safety can be improved through the collection and analysis of data gathered from various sources, such as inspection, sensing, and monitoring.

The aim of this paper is to show the software development approach of a total system health management support web application to aid in decision-making and planning by the pipeline operators. The software platform is a prototype dynamic heath monitoring of the pipeline systems.

2. SOFTWARE PLATFORM REQUIREMENTS

The goal of the software platform is to integrate the data, methods, models, and technologies into a total system health management support tool to aid in decision-making and planning by the pipeline operators. The foundations and operational concepts are depicted in Figure 1.

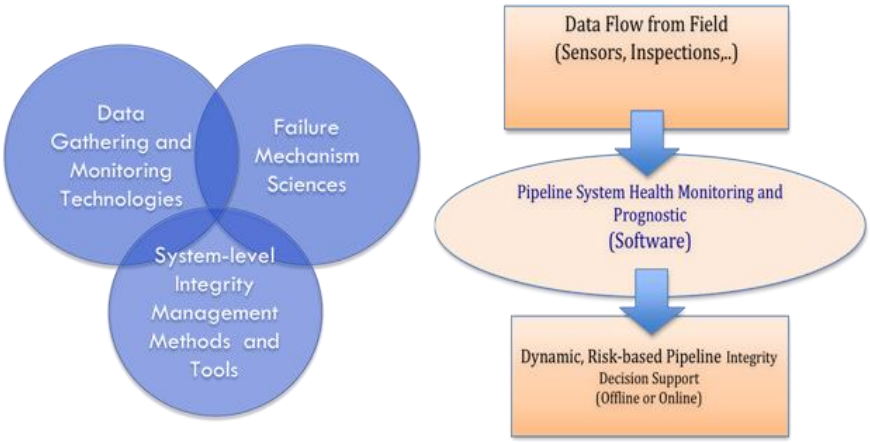

Figure 1 Foundations and Operational Concept

* Corresponding author's email address: diacon@ucla.edu.

The following are examples of technologies that can be integrated:

- Intelligent Pigging for Persistent Pipeline Health Monitoring;
- Sensing capabilities for online monitoring of various parameters;
- Decentralized Supervisory Control and Data Acquisition System (SCADA) for Pipeline Leakage Estimation;
- Data gathering for the development of corrosion failure mechanisms.

The pipeline health monitoring system software platform will include:

- Integrity assessment based on comprehensive range of evidence from sensing, inspection, and monitoring in addition to probabilistic integration of mechanistic models and data on failure mechanisms relating to various causal factors (e.g., material, environment, manufacturing, etc.) for assessment of the pipe segment health (remaining life);
- Dynamic pipeline network probabilistic health assessment model software for optimal risk-based prioritization of inspection and proactive management;
- Geographical mapping capabilities that will augment the interaction of the pipeline operators with the pipeline system.

An example of the flow of analysis from information to decision support is shown in Figure 2.

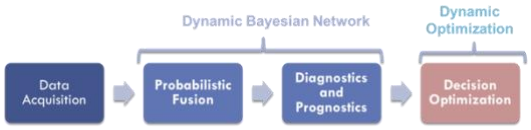

Figure 2 Analysis and Decision Support Framework

In Figure 3 an information dashboard that will be accessible to the operators in the control room is shown.

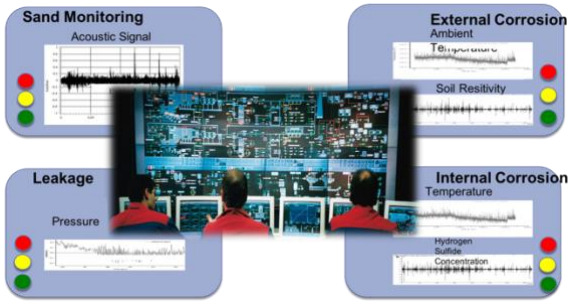

Figure 3 Risk-based Dynamic Pipeline Integrity Management System

The output of the pipeline health monitoring system software platform should give online or offline updates on the reliability state of various segments of the pipeline system, and dynamically update the suggestions on when and where to take mitigating actions, e.g., increase or decrease inspection frequency, based on total ownership cost concept such one shown in Figure 4.

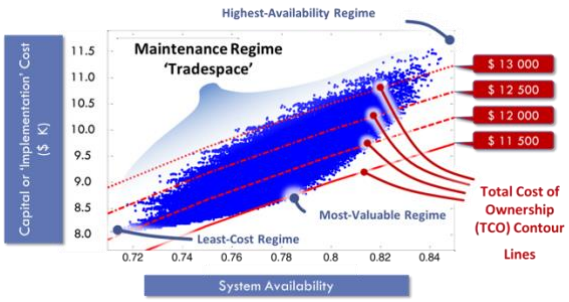

Figure 4 Dynamic Maintenance and Inspection Optimization, based on "total cost of ownership" concept.

3. SOFTWARE PLATFORM ARCHITECTURE

In the early stages of the software platform architecture design phase, two alternatives were considered:

- Adopt an existing system reliability and risk modeling platform known as Information Risk Information System (IRIS), and build new features needed for Pipeline Health Monitoring,
- Design a new software platform that would give greater flexibility in the employed software development languages and tools (e.g., IRIS).

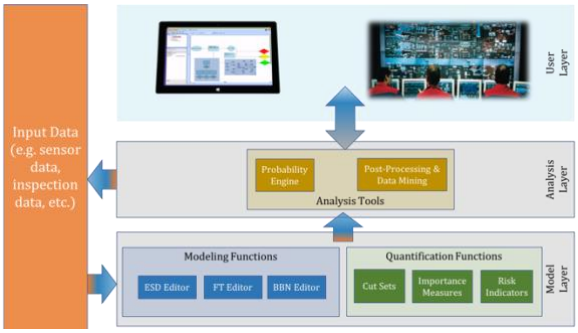

Figure 5. Pipeline Integrity Management System Design Based on the IRIS Platform

The second option was selected (Fig. 5). From a functional point of view, it contains three layers: Model Layer, Analysis Layer, and User Layer. The Model Layer will be supported by the already existing system reliability and risk features (i.e., Fault Trees and Bayesian Belief Networks) of the current IRIS platform (Wang, 2007). Most of the new features (for use in pipeline system operating environment) that would need to be added will support the Analysis Layer, and the User Layer. Namely, interfaces to the new computational tools listed in the requirements section will be added to the Analysis Layer and will also support the User Layer containing the dashboard for user interaction and monitoring.

The Modeling Layer is currently being developed in the form of a cross-compatible Web Application called Hybrid Causal Logic Analyzer shown in Figure 6. The files defining the diagrams for ESDs, FTs, and BBNs follow the Open-PSA Model Exchange Format 2.0 that will provide flexibility with other system modeling platforms.

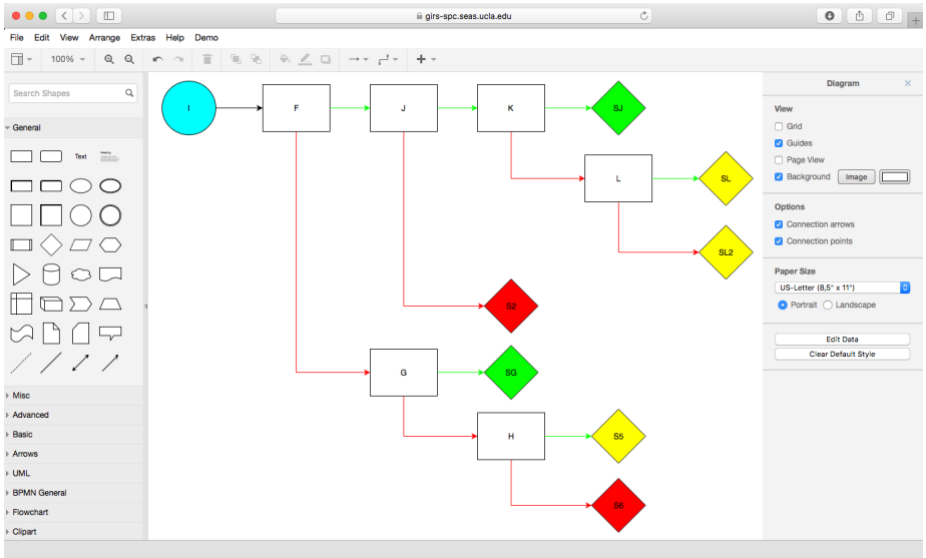

Figure 6 Model-Building Layer being developed in the form of a cross-compatible Web Application

The Analysis Layer and User Layer will be based on a map stack software architecture. This is a common online software architecture used for mapping applications in which users have seamless

access to a click-and-drag geographical map with selective information layers showing various information (i.e., street names, infrastructure networks, etc.) As is shown in Figure 7, the map stack is made of four major elements: browser UI, tile cache, map server, and geospatial data.

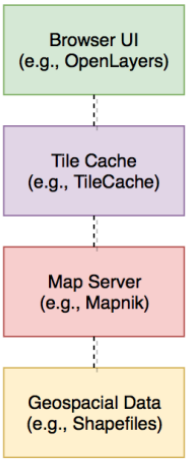

Figure 7 Web Application Architecture

In the applications built using the map stack architecture, users interact with it through a JavaScript or Python library that continuously performs a series of tasks. The typical ones would be to listen to user events, request tiles from the map server, assemble them in the user interface (UI), and overlay additional elements on the map (e.g., vector shapes).

A high-performance web mapping library written in JavaScript under an open source license is OpenLayers. It is a very popular library that supports event handling, the reading and writing of a wide range of geospatial data formats, and vector data drawing. Through its flexible API it can be linked to multiple map server backends.

A tile cache is a server that is placed between the browser and the map server. For each map tile request, it checks to see if it already exists in a cache, from where it can be instantly pulled without calling the map server to generate it. If the map tile is not already created, the tile cache requests it from the map server and also saves it to speed up subsequent requests. For the most part, this layer is transparent to the user and is typically a simple set up and configuration. The client is pointed to the tile cache, and the tile cache is pointed at the map server. The tile cache is an optional element in the map stack, because it is not necessarily required in order to process user requests for a tile from the map server. However, in practical use cases, web applications are under loads that make continuous map tile generation expensive and wasteful. The tile caching technology is the element that makes open source web mapping a viable alternative. TileCache is an open source tile caching server that is relatively simple to set up. It can store tiles on local disks, private servers, in Memcached, or on Amazon's S3 storage service.

A map server is an application that takes geospatial data as input and renders graphical output. Specifically, for modern online mapping applications, it generates a series of map tiles, which are uniformly sized graphic files (e.g., 256×256 PNGs) that are ultimately served to and assembled in the browser UI as the displayed map. The map server handles the final appearance of the maps by styling the geospatial elements, in a manner similar to Cascading Style Sheets (CSS) language. Essentially, most of the map servers provide features for defining various styles (e.g., stroke width and color, fill color, opacity, and layer stacking order). In some applications, the map servers can be open directly to the internet, where a well-defined interface takes secure HTTPS requests representing specific geographic locations and returns the corresponding tiles. Nevertheless, for modularity, it is good practice to hide the map servers from being accessed directly by the browser UI by interleaving the tile

cache layer in the map stack. Mapnik is one example of a popular open source map server that has a developer-friendly interface and, most importantly, it renders efficiently.

The building blocks of any mapping application are the databases that define the points, lines, and polygons that represent real-world places, streets, and areas. Geospatial data connects geometric shapes with a coordinate system (e.g., latitude and longitude) and with attribute or feature data such as location names or highway designations. The big element in the geospatial data world is a database of streets. While not every mapping application needs a layer with streets, most do, and users expect it as a primary way to orient themselves. However, for the pipeline health monitoring system adding a pipelines database is a key ingredient. Moreover, the seamless interfacing of advanced quantification techniques is critical to a support tool decision-making and planning by the pipeline operators.

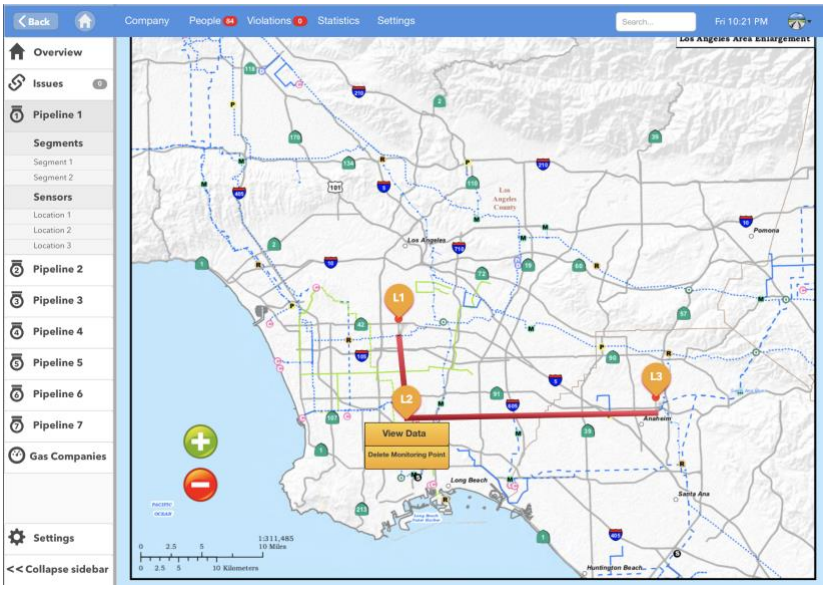

Figure 8 Prototype of the Pipeline Health Monitoring and Management Web Platform

4. CONCLUSION

The pipeline health monitoring system software platform under development provides ability for (a) Integrity assessment based on comprehensive range of evidence from sensing, inspection, and monitoring in addition to probabilistic integration of mechanistic models and data on failure mechanisms relating to various causal factors (e.g., material, environment, manufacturing, etc.) for assessment of the pipe segment health (remaining life); (b) Dynamic pipeline network probabilistic health assessment model software for optimal risk-based prioritization of inspection and proactive management, and (c) Geographical mapping capabilities that will augment the interaction of the pipeline operators with the pipeline system. The selected architecture foundation for the software is the IRIS platform with wide range of capabilities in terms of system model building and probabilistic analysis of various types of evidence for assessment of model parameters. On this foundation we are adding capabilities to integrate predictive corrosion models, third party damage models, natural hazards models, and computational modules to perform pipeline health assessment, and develop inspection and maintenance strategies. The resulting software will be deployed as a control room health dashboard and hand-held web-based field inspection support tool.

Acknowledgements

The research leading to development of this paper was supported by Pipeline System Integrity Management research project sponsored by the Petroleum Institute, Abu Dhabi, UAE through a sub-award from the University of Maryland Department of Mechanical Engineering.

References

[1] M. Pourali, A. Mosleh, A Bayesian Approach to Online System Health Monitoring, Reliability and Maintainability Annual Symposium RAMS, pages: 210-215, (2013).
[2] M. Pourali, A. Mosleh, A Functional Sensor Placement Optimization Method for Power Systems Health Monitoring, 2012 IEEE Industry Application Society Annual Meeting, (2012).
[3] M. Pourali, A. Mosleh, Application of Bayesian Sensor Placement Optimization for Real-Time Health Monitoring, ASME International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE), (2012).
[4] E. Rabiei, E. L. Droguett, M. Modarres, A Prognostics Approach Based on the Evolution of Damage Precursors Using Dynamic Bayesian Networks. Advances in Mechanical Engineering 8, no. 9: 1687814016666747, https://doi.org/10.1177/1687814016666747, (2016).
[5] C. Wang, Hybrid Causal Logic Methodology for Risk Assessment. University of Maryland, (2007).