INL/MIS-22-69294-Revision-0

The DeepLynx Data Warehouse

September 2022

John Wayne Darrington

Changing the World's Energy Future

INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance, LLC

INL/MIS-22-69294-Revision-0

The DeepLynx Data Warehouse

John Wayne Darrington

September 2022

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

The DeepLynx Data Warehouse

By John Darrington

Introduction

Digital Engineering and the development of digital twins necessitate the use of highly sophisticated tools and software. These methodologies and systems bring many benefits but rely on accomplishing challenging goals such as disparate data and systems integration into a single cohesive and holistic system.

Idaho National Laboratory has recently released an open-source data warehouse – DeepLynx – to help in the planning, creation, execution, and management of projects in the digital engineering space, with an emphasis on supporting digital twins in an operational space.

Terms and Foundational Knowledge

To understand this document fully, we encourage the reader to use this section as a reference tool so that as we refer to the many aspects of this work, you may be confident in the understanding and knowledge imparted by this document. The following terms and definitions will help you achieve this.

Data Warehouse

"A data warehouse is a type of data management system that is designed to enable and support business intelligence (BI) activities, especially analytics. Data warehouses are solely intended to perform queries and analysis and often contain large amounts of historical data."[1]

Data warehouses, particularly when used for operational purposes such as a digital twin, are designed to house important operation and modeling information, and then disseminate the information to various programs, machine learning models, and people for analysis and study.

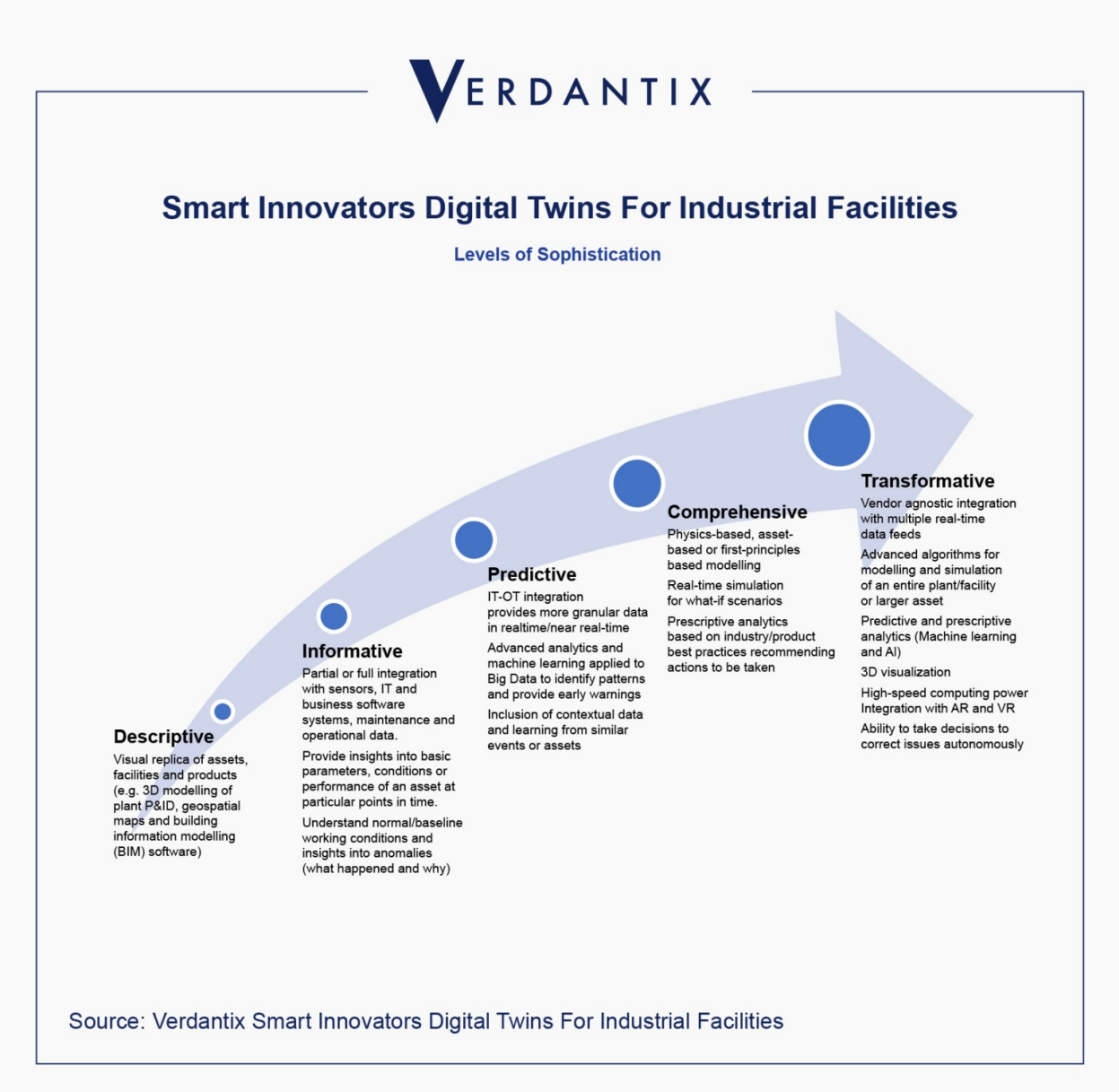
Digital Twins

A basic definition of a digital twin is a system which provides a real-time digital representation of some physical asset or process of interest, with the purpose of improving performance and efficiency. The concept of a digital twin is relatively new and a single, unified view on what a digital twin is, still has not been achieved by industry. In general, a digital twin is thought to have three parts including a physical asset or product, a virtual representation of that product, and a bi-directional data connection between the physical and virtual. With the application of various types of analytics to the physical data, it is expected that a digital twin would create an

[1] https://www.oracle.com/in/database/what-is-a-data-warehouse/

evolving representation of the physical process and help in optimization of the process over time.

A digital twin may have many levels of completeness or operational capability. Below is a graphic from Verdantix, illustrating what Digital Engineering at INL has adopted as a basis for our own classification of digital twins.



2

[2] https://www.verdantix.com/report/smart-innovators-digital-twins-for-industrial-facilities-2021

Ontologies

"A body of formally represented knowledge is based on a conceptualization: the objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them (Genesereth & Nilsson, 1987) . A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly.

An ontology is an explicit specification of a conceptualization."[3]

In briefer terms, an ontology in the context of DeepLynx and digital twins is a data model which allows us to name, classify, and apply specific properties to the various concepts which will be represented in the system as well as define and classify the relationships between them.

What is DeepLynx?

DeepLynx is an ontological data warehouse with timeseries data support written in Node.js[4] and Rust[5]. DeepLynx is backed by PostgreSQL[6] and other open-source software. DeepLynx is open-sourced and is focused on enabling complex projects to embrace digital engineering. It allows projects to realize the digital thread and systems such as digital twins through integrations to a large collection of software systems across a project's lifecycle. DeepLynx allows users to ingest or fetch data from disparate sources and combine them in an easy to view and navigate 2D-graph. This graph is a collection of nodes and edges, where each node represents a data point, and the edges relate the data. This graph follows the dynamic data model defined by the user in their ontology. DeepLynx also enables users to perform analytics on retrieved data in a unified manner, using the user-defined ontology for naming and classification.

Apart from its data warehousing capabilities, DeepLynx also contains various strategies for storing the raw files that such data might come from. This might include files such as images, videos, or other assets. These assets can be attached to data inside the graph, allowing users to quickly retrieve the relevant assets for any given piece of data.

DeepLynx also supports the storage of timeseries data in large quantities, a necessity when supporting a digital twin in an operational capacity. Timeseries data is enabled using the open-source Postgres plugin TimescaleDB[7]. More information can be found in "Technology Behind DeepLynx" later in this document.

---

[3] http://www-ksl.stanford.edu/kst/what-is-an-ontology.html
[4] https://nodejs.org/en/
[5] https://www.rust-lang.org/
[6] https://www.postgresql.org/
[7] https://www.timescale.com/

Communication with DeepLynx takes place primarily via a REST API – communicating over HTTP and serializing/deserializing primarily JSON formatted data. DeepLynx also supports ingestion of CSV and XML documents. The open-source release also contains a bundled web application, giving users a Graphical User Interface (GUI) for interacting with DeepLynx's API layer. DeepLynx also serves as a basic, oAuth2[8] compliant identity provider – allowing you to develop your own application powered by DeepLynx but branded the way you wish for your customer or use case.

9

Brief History of DeepLynx

DeepLynx has been in active development since 2019 and was released under the MIT license[10] as open-source software on June 29th, 2020[11]. DeepLynx's primary authors are John Darrington and Christopher Ritter.

Originally designed to tackle Model-Based Systems Engineering (MBSE) tool integrations and warehousing, DeepLynx has evolved to handle several use cases including support for operational use through systems like digital twins. Most recently DeepLynx has stepped into the role of operational data provider with the addition of timeseries data storage support.

DeepLynx continues to be actively developed by a team based at the Idaho National Laboratory and as part of the DICE Digital Thread Group[12].

Use Case: Reactor Digital Twin

For this section we will use a fictional digital twin and discuss how DeepLynx enables the digital twin to reach the highest levels of the earlier discussed digital twin levels. We will examine DeepLynx's role from start to finish and how you can benefit from using it at the earliest stages of a digital twin.

---

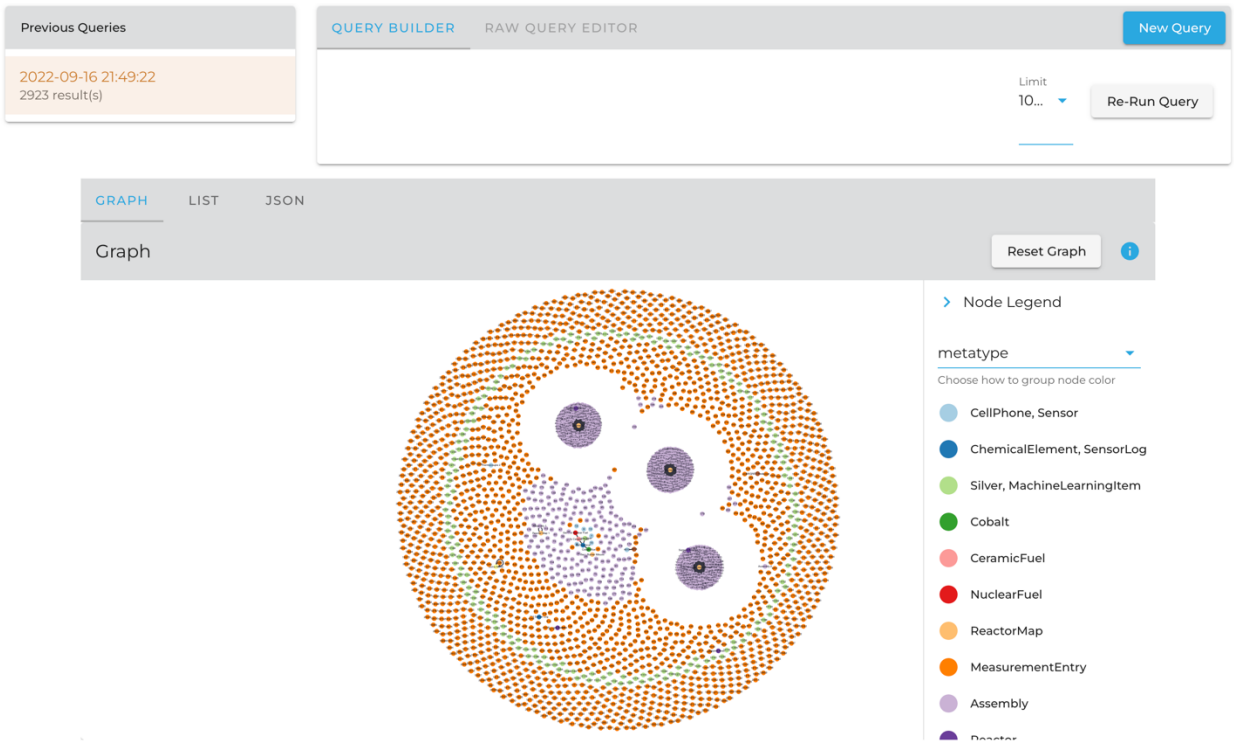
[8] https://oauth.net/2/
[9] DeepLynx Graphic
[10] https://github.com/idaholab/Deep-Lynx/blob/master/LICENSE
[11] https://github.com/idaholab/Deep-Lynx/commit/fcadc5362e8dfe697087369c1f638e4dca72ca1e
[12] https://dice.inl.gov/digital-thread/

Level 1: Descriptive

While DeepLynx itself is not a 3D-Modeling tool, work has been done to integrate it with programs such as Aveva Everything3D. By analyzing and ingesting model information – such as equipment definitions and properties, DeepLynx can begin to build a digital representation of a physical asset. Such a representation might take the form of models displayed in a program such as Unity [13]or a tool like the Microsoft HoloLens.

In the design stage, requirements detailing the proposed facility, reactor, or item that DeepLynx is representing are extremely important to all those involved in the project. DeepLynx possesses the capability to integrate with requirement management programs – such as Doors Next Generation (DNG/JAZZ) – and then assign those requirements to the 3D model information already ingested.

DeepLynx's intuitive and unique type mapping system allows users to not only choose how their ingested data is represented under the defined ontology, but also how that data relates to itself and data from other sources. In this case the system enables drawing connections between 3D model data and requirements and displaying that connection to a user via a Graph interface.



14

[13] https://unity.com/

[14] DeepLynx Graph View via the Graphical User Interface (GUI) representing a reactor and it's control assemblies.

Level 2: Informative

Once a physical asset is digitally modeled in DeepLynx, users can convert any record, or node in the graph, to become a timeseries data "bucket" node. For example, a user might convert a node whose type is "Sensor" and which has a unique identifier matching an actual sensor in the physical asset, into a timeseries "bucket". This "bucket" allows users to directly store the sensor's information on the graph, attaching it to the individual node. The only limit on how many timeseries data "bucket" nodes you can have is how quickly the hardware running DeepLynx can ingest the data at your required intervals. Work is being done to make DeepLynx capable of ingesting many millions of records each second.

Level 3: Predictive

Now that DeepLynx is storing your timeseries data in a manner representing the physical asset, you can then query said timeseries data. We provide a GraphQL enabled REST endpoint to allow users to quickly and intuitively query both data on the graph and in timeseries data buckets. This query system is the basis for powering machine learning adapters and integrations.

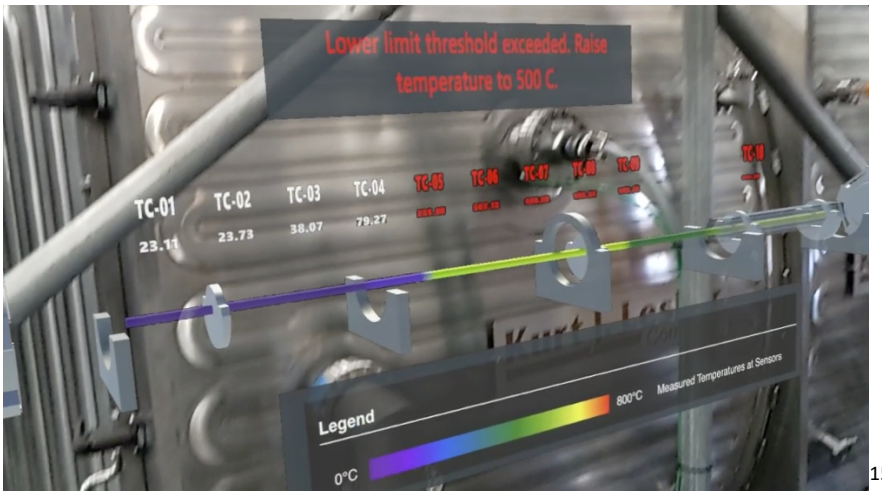
In a level 3 twin, DeepLynx provides machine learning integrations and scripts with an easy way to retrieve data to run their predictions. In many instances, DeepLynx has also served as the storage medium for the results of said predictions – meaning the data comes full circle. Work is being done to also enable users to perform model creation and prediction directly from DeepLynx and its database.

Level 4-5: Comprehensive and Transformative

How DeepLynx supports level 4 and 5 digital twins is very similar to and builds off of what was previously discussed. As a receptacle for both the raw data and data that's been run through various machine learning models, DeepLynx can serve as a basis for real-time modeling and simulation as well as 3D visualization tools.

A Transformative digital twin utilizes bi-directional communication with the physical asset to perform some sort of input or control, such as optimization controls that may be related to the data or physics-based predictions included in the digital twin. DeepLynx and its integrations can enable this final digital twin level.

Work has been done to integrate DeepLynx with augmented reality (AR) devices such as the Microsoft HoloLens. These integrations allow the user to visualize the data, perhaps overlaid on a 3D representation of the physical asset or the physical asset itself.



[15]

Benefits of Choosing DeepLynx

There are many benefits for choosing DeepLynx to support your work in the digital engineering space. Below are just a few which we feel are the most impactful.

- **Open Source:** DeepLynx is completely open source, including its published adapters and integrations to its included web interface. It can be configured to use only open-source tools under the hood such as PostgreSQL and RabbitMQ. This enables teams to deploy DeepLynx at very little to no licensing costs. This enables users to only pay for the hardware for DeepLynx to run on. DeepLynx can save an organization hundreds of thousands of dollars in licensing that off the shelf software might require to accomplish the same goal.
- **Easy to Learn:** While data warehousing can be a complicated task, we hope that DeepLynx allows users to process and analyze their data more quickly by streamlining the data ingestion and mapping process. We try to make it as simple as possible to get your data into and out of DeepLynx and maintain its integrity in the process.
- **Versioning:** Both user-defined ontology and raw data (apart from timeseries data) is fully versioned in the system. That means you can confidently make and track changes to the user ontology and data without having to worry about irreparably damaging it. We use a ledger system to keep track of your raw data – meaning that along with the ontology versioning we can potentially show you how your data looked (and was mapped) at any point in the past.
- **Reusable Ecosystem:** DeepLynx provides many open-source adapters that can be used as a starting point from which to build your integrations. These adapters provide functionality around integrations with DeepLynx and a specific application or domain, allowing you to hit the ground running and focus on your specific data and needs.

[15] DeepLynx powered Augmented Reality View of MAGNET

DeepLynx Future Roadmap

While DeepLynx currently offers many features and benefits for a digital twin ecosystem, additional features are planned and in-progress that will allow it to become an even greater asset for enabling digital engineering and developing digital twins. These features are not set in stone and should not be treated as an official roadmap. Also, they are not listed in the order in which they will necessarily be accomplished.

Multi-Node Support

While the DeepLynx application itself is designed to scale horizontally, this scaling does not currently affect the PostgreSQL database instance that DeepLynx communicates with. While we don't foresee an instance in which the graph storage of DeepLynx will need to span multiple nodes (or machines), timeseries data capture is another matter. To support timeseries data ingestion of up to terabytes of data daily, we need to be sure a strategy and features are in place to allow the scaling of our backing PostgreSQL database across multiple different nodes (or machines).

Our current plan for supporting large timeseries data ingestion involves the usage of TimescaleDB's multi-node environment[16]. This built-in feature will allow us to create multiple instances of a PostgreSQL database across different machines and then connect them altogether into a single cluster for timeseries data storage. This is how we will be able to store the potential petabytes of data expected in future projects.

Citus was originally looked at for a multi-node PostgreSQL instance and can still be considered. However, it is not compatible with TimescaleDB. Work can be done to make sure they play nicely together, but again – we don't foresee an instance in which the ontology or even nodes/edges will need to be distributed.

Data Partitioning/Security

Due to the potentially classified nature of data being stored inside DeepLynx, work needs to be done with regards to data security and partitioning. We need the ability for users to specify subsets of data inside their containers so that they can then assign permissions to view and edit data based on those subsets. This will allow users to control access more accurately to potentially sensitive data and help maintain overall security. Work also must be done on general DeepLynx security so that we can be confident in the safe storage of a user's data.

Reports Management, Scheduling, Hooks, and Download

While DeepLynx has the ability for users to query data stored inside its database, users must currently run these queries manually through the API, or simpler queries through the provided GUI. This presents numerous limitations, some of the most important being the ability to return large amounts of data to the end user in a timely manner. While the ability to download all results from a query to a file exists currently, this isn't well known and is still only accessible

[16] https://docs.timescale.com/timescaledb/latest/how-to-guides/multinode-timescaledb/

through the manual endpoint. We propose a reports management layer to hopefully alleviate these problems.

Unity WebGL/Model Management

Due to the continued work with Unity and mixed reality, we propose that DeepLynx become a way to manage and host WebGL builds or models and represent node/edge and timeseries data through these models. The ability for DeepLynx to serve a WebGL build, for example, has already been proven – meaning that DeepLynx could become not only the repository for data regarding a project but also the mixed reality assets representing it.

Some work to support this has already been accomplished, such as blob storage through various providers and serving as an asset store.

Deployment and Production Services

DeepLynx is a complicated program and consists of many moving parts. It can be daunting for those simply wanting to experiment in the system or use it for a small project. While a Docker Compose file and ecosystem exists, it's only acceptable in a development environment. We currently do not have any production level deployment instructions, scripts, or environments. Work must be done to make deployment of a production ready DeepLynx cluster as painless and easy as possible, as well as ways to manage the data storage and scaling of the tool.

Conclusion

DeepLynx has the capability to be a powerful tool in the digital twin space. While still in active development, it has already proven its usefulness in various projects and situations. With the support of a user-defined ontology and clever use of existing technology, DeepLynx has the potential to be a driving force behind modern digital twins.