Proceedings of the Design Society, Volume 6: DESIGN 2026
https://doi.org/10.1017/pds.2026.10429

Designing digital twins: a graph-based schema to enable systematic cost-benefit analysis

Adam McClenaghan✉, James Gopsill and Ben Hicks

University of Bristol, United Kingdom

✉ adam.mcclenaghan@bristol.ac.uk

ABSTRACT: This paper posits that a Digital Twin can be viewed as a collection of nodes and edges where nodes represent actions on/with data and edges represent the flow of data between nodes. The paper provides a schema whereby the nodes and edges can be defined and the costs and benefits can be attributed, as well as analysis techniques enabled by the schema. The potential of the schema in supporting the design of Digital Twins is then demonstrated through a worked example, in which it is shown that traditional bottom-up cost estimates significantly overestimate costs when compared with this approach.

KEYWORDS: digital twin, cost management, industry 4.0, cost-benefit

1. Introduction and background

Following Grieves (2015) seminal whitepaper describing the Digital Twin (DT) Academic publications on DTs have grown at an average rate of 47% per year between 2015 and 2025 (Scopus). This academic enthusiasm is mirrored in market predictions. The Digital Twin industry is expected to grow from a current valuation of £14.38B to £210.36B by 2032, a compound annual growth rate of 39.8% (Insights, 2025). Interest from technical leads in industry share this sentiment with 59% of executives across various industries planning to integrate Digital Twins (Businesswire, 2023), while 70% of c-suite technology executives are actively exploring or investing in Digital Twin technology (Mckinsey, 2022). A 2024 industry study surveyed 660 C-level executives across 11 industries and found that 91% of respondents were planning to maintain or increase their investment in Digital Twins in the coming year, and 72% were planning to increase their spending in this area (Hexagon, 2024). In the same study, 75% of respondents who claimed that they were very familiar with Digital Twins thought that they could deliver significant value for their organisation. Despite the enthusiasm and market expectancy, there exist few methodologies and tools to support firms in designing their Digital Twin and subsequent investment strategy to realise it. The majority of support to date comes through consultancy projects with DT experts. Cost estimate heuristics vary widely during the concept design stage, highlighting the financial challenge. The development of the core Digital Twin software platform alone, not including hardware, data connections, or simulation models, is estimated to be $20,000 to $45,000 for a small company, rising to between $75,000 and $90,000 for large enterprises (RisingMax, 2024). More holistic estimates, which include all elements (sensing, communication, simulation, intervention) of the DT, range from $300,000 for a Digital Twin of a distribution centre, to between $4 million and $5.6 million for a complex physical entity such as a shopping centre (Lengthorn, n.d.). It can be particularly difficult to estimate the total cost of a Digital Twin project at the early conceptual design stages leading to uncertainty around investment, resulting in delayed or slow adoption (Mihai et al., 2022).

Traditional methods for undertaking investment decisions, are not sufficient to understand the cost-benefit proposition of DTs. Techniques such as Net Present Value or Internal Rate of Return typically treat the project as a monolithic capital expenditure (Capex) with predictable returns (Brealey, 2023).

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press

These approaches fail to capture the emergent nature of the return of a DT, and the interconnected nature of the cost drivers of the many assets that form a DT and the benefits they provide and contribute to. Analysis of the economics of DTs highlights that characterising the full spectrum of costs and benefits is a significant challenge, with benefits often embedded in operational processes, making it difficult to assign value (Thomas, 2024). The constituent elements of a DT are a mixture of Capital Expenditures (CapEx), and ongoing Operational Expenditure (OpEx) costs, which can vary greatly, and scale non-linearly with the number of assets, data fidelity and model complexity (McClenaghan et al., 2023).
DTs are emergent in nature, they can be implemented over time with new sensors, databases, and feedback mechanisms being added. At the early concept stage, it is beneficial to be able to map out the entire space, which can then be queried to define the elements that offer the greatest cost-benefit proposition to the company.
This paper contributes to the design of Digital Twins by providing a schema that augments existing DT network architecture modelling practices with the cost information necessary to evaluate costs at the concept stage. The paper continues by summarising related work, including existing cost estimation techniques that serve as inputs to any financial model and the uses of graphs in engineering in Section 2, followed by presenting the schema and the methodology used to derive it in Section 3. Analysis metrics for the schema are presented in Section 4, with a worked example demonstrating their use shown in Section 5. Finally, Section 6 discusses the implications, advantages, and limitations of this approach, concluding with directions for future work.

2. Related work

This section presents existing costing methodologies, and the drawbacks of each. It then follows with an examination of how graphs are used in the engineering field to represent systems before introducing schemas, and the benefits they provide when applied to graphs of this nature.

2.1. Costing methodologies

There are three main categories of cost estimation techniques, analogous (Top down) methods, parametric methods, and bottom up methods, also called "grassroots" methods (NASA, 2023). Table 1 provides a description, and a summary of the benefits and drawbacks of each technique.

Table 1. Overview of costing techniques

Costing technique	Description	Benefits	Drawbacks
Top-down	Takes costs of historical projects and adjusts them to the current project based on expert analysis or other techniques	- Fast and inexpensive - Simple; easy to understand and apply - Good for ball park estimates	-Least accurate -Lacks detail Requires analogous projects -Subjective
Parametric	Uses statistical relationships between historic project costs and project parameters to estimate future costs	- Can be more accurate than analogous - Objective; based on statistical models - Scalable: can be applied to any project size	-Requires good quality data from previous project -Not suitable for unique projects - Statistical relationships could be complex/coupled
Bottom up	Involves breaking down the project into constituent parts, estimating the cost of each individually, then totalling the estimates of these parts	-Most accurate -Allows parts of project to be costed individually - Improves resource and schedule planning	- Most time consuming - Requires full project scope - Cost estimates for each constituent part prone to inaccuracy

This bottom up approach is well suited to the modular nature of a Digital Twin, however a simple aggregation of parts fails to capture the interdependencies and non-linear scaling effects inherent in DTs. To extend this bottom-up approach and enable a full cost-benefit analysis (CBA), this paper proposes that the costs and benefits can be represented through a directed, attributed graph.

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press

2.2. Graphs in the design of engineering systems and digital twins

A DT is a system, a combination of interacting elements organized to achieve one or more stated outcomes, (International Organization for Standardization et al., 2023). An intuitive way of modelling systems is a graph consisting of nodes that represent system elements and edges representing the relationships between them (Heckmann et al., 2014). The use of graphs for the representation of engineering systems is a mature area of research, with applications in fields such as electrical circuit analysis, modelling transport systems, and signal flow, among many others (Deo, 1974). In the costing space, the benefits of graphs are being increasingly utilised; Khan et al. (2024) used them to calculate and optimise the cost of cloud infrastructure, while Guo et al. (2018) utilised graphs to model the costs of changes to supply chains.

To effectively represent Digital Twins in this way, a directed multi-graph is required, which has a direction constraint placed on edges, and multiple possible edges between nodes. More detail of the mathematical foundations of graphs of this type can be found in (Diestel, 2018).

Graphs of engineering systems typically adhere to defined schemas that enable analysis of the complex technical correlations within a system, and efficient storage in databases (Patzwahl & Hyba, 2022). A schema is a set of rules that governs how data is organised, named, and how different pieces of data relate to one another. Examples include JSON (Pezoa et al., 2016), GraphML (Brandes et al., 2002), GML (Himsolt, 2010). Schemas such as these enables data consistency, validity and interoperability for data. In contrast, architecture diagrams for DTs at the conceptual stage remain largely visual with ad-hoc unstructured annotations. It is often only at the embodiment design stage where more structured schemas, such as UML, are used to define the interfaces between assets within a DT.

3. Schema

This section details the proposed cost schema to augment existing DT architectural network design practices with cost information in a structured manner that can be subsequently used to provide insights and forecast the cost of a DT deployment earlier in the design stage.

3.1. Schema development

In literature, DTs are typically presented in the form of an architecture diagram that depicts the key services and data flow(s). To translate these architectures to graphs representations that can be used for cost analysis, a schema is required to govern the types of nodes, edges, and required parameters for each. To establish the requirements for the schema, we searched Scopus for publications that contained holistic Digital Twin architectures. The search resulted in 10 papers that presented complete representations of their architecture. We then extracted the necessary detail and synthesised them into simplified representations, containing all nodes, and representative edges connecting them. Two examples of the outputs of this process can be seen in Figure 1. From these graphs, we derived an initial schema containing the minimum set of node types that could represent the graphs. Through a series of iterations, we refined the schema until it captured all the necessary information and could represent all the presented architectures completely. In the remainder of Section 3 we define the final schema produced in this process.

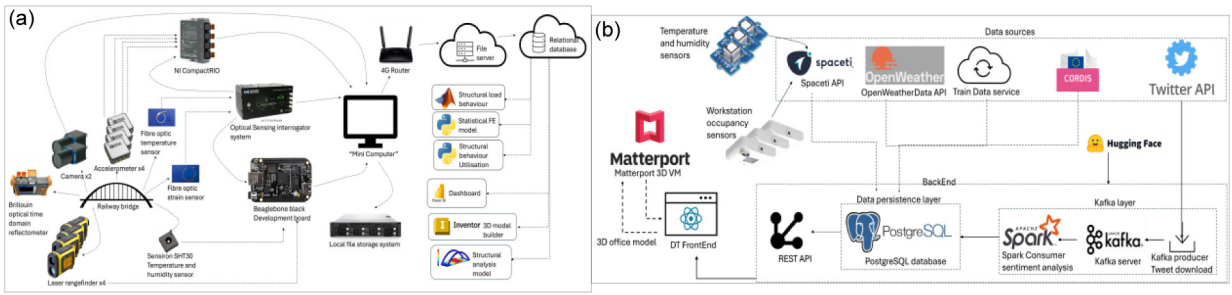


Figure 1. Examples of adaptations of architectures from literature, (a) adapted from Gürdür Broo et al. (2022), (b) adapted from Alonso et al. (2024)

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press

3.2. Node schema

Through the iteration of the schema, we determined that all architectures could be represented by a graph containing four node types: 1) Source; 2) Sink; 3) Store, and, 4) Transformer.
All four node types inherit a base node definition (Figure 2(a)), which describes attributes that all nodes have in common. Every node has an *id*, an optional *name* and *description*, an *active* Boolean and an associated cost function. The *active* Boolean determines whether that node is included in the costing calculations. This is utilised when isolating data streams to examine the cost-benefit proposition of different Digital Twin outputs. One of the key findings from the review of architectures and cost elements is that the costs can all be modelled by a function, parameterised by the size and frequency of data entering the node. This cost function also encapsulates the costs incurred by transferring data between nodes. The schema places no constraints on the structure of the cost function as it will be dependent on the context that node represents, however it will be a function of the size and frequency of the data flowing into the node and will return data in the form of a *Cost,* the definition of which is shown in Figure 2(b). The function will have access to the node's context and must be defined.

3.2.1. Source

A source is a point at which data enters the DT. It is defined in the schema as a node that has a data output, but no data input. Examples of sources could include sensors, connections to external APIs to get information, human input from a technician, as well as data from a manufacturing machine such as a 3D printer. The class definition for a source can be seen in Figure 2(c). The *send* attribute is a list, to allow the source to send data to multiple nodes, to send multiple pieces of data to one node, or a combination of the two. The one-to-many mapping of data out to another node would allow for the situation where a sensor is producing multiple data types, as each data edge specifies one piece of data. An example of this would be an accelerometer, sending x, y, z data. In this scenario, the accelerometer would be represented as a single source, however each piece of data out would be carried along its own edge to its respective destination.

3.2.2. Sink

A sink is a point at which data exits the DT and provides a useful output outside of the DT, allowing value to be derived from the data that enters the sink. It represents an output of the graph and is defined as a node that receives information, does not produce any, and enables the realisation of a benefit from the information it receives. Examples of sinks could be a dashboard, an email notification service, or an automated intervention that happens based on the data it receives. The sink class includes a benefits attribute, which can be used to assign benefits to the sink, such as a percentage throughput improvement, or a reduction in the number of defective parts. The class definition of a sink can be seen in Figure 2(d).

3.2.3. Store

A store node is defined as a node that both receives and produces data but does not manipulate the data. The cost function of the Store node captures the cost of holding data in that node. The send and receive attributes contain lists that define the edges into and out of each node. The class definition of a Store can be seen in Figure 2(e). Examples of stores include Data Lakes, Databases and Blob Stores.

3.2.4. Transformer

A transformer is any node that receives and produces data and performs some manipulation on that data. The manipulation performed is not defined in the schema at this stage, but the cost of the manipulation is captured in the cost function, inherited from the base node class. Examples of transformers include a simulation running in a cloud based virtual machine, or a serverless cloud computation instance that is activated to clean data before it is passed to another element. Each Transformer has an attribute for *send* and *receive*, each containing a list of all the edges into and out of the node. The class definition can be seen in Figure 2(f).

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press

```python
class BaseNode(TypedDict):
    id: str
    name: Optional[str]
    description: Optional[str]
    active: bool
    cost: Callable[[], Cost]
```
(a)

```python
class Cost(NamedTuple):
    amount: float
    unit: str
```
(b)

```python
class Source(TypedDict, BaseNode):
    send: List[DataEdge]
```
(c)

```python
class Sink(TypedDict, BaseNode):
    receive: List[DataEdge]
    benefits: List[BenefitEdge]
```
(d)

```python
class Store(TypedDict, BaseNode):
    send: List[DataEdge]
    receive: List[DataEdge]
```
(e)

```python
class Transformer(TypedDict, BaseNode):
    send: List[DataEdge]
    receive: List[DataEdge]
```
(f)

Figure 2. Base node and cost (a,b) and node definitions (c-f)

3.3. Edge schema

With the nodes defined, the schema must now define the connections between them. In the same way as the node schema, the edge schema begins with a base edge class, as can be seen in Figure 3(a). The base edge contains an *id* attribute, an *active* attribute, which behaves in the same way as this attribute of Base Node, and *u* and *v* attributes, which define the start and end nodes respectively.
Inheriting from the Base Edge class is the Data Edge.

3.3.1. Data edge

The data edge defines a connection where data is being transferred between two nodes. The class definition can be seen in Figure 3(b). Since DataEdge inherits its start and end nodes from the BaseEdge class, the only attribute is data, which defines the data passed along the edge. This data is governed by the Data class, as can be seen in Figure 3(c). The data class contains an *id* attribute, to allow that piece of data to be traced throughout the graph, as well as optional *name* and *description* attributes. It also includes metadata describing the underlying data; its *frequency*, how often that piece of data is transferred along that edge, and the *size*, the size of each piece of data being transferred. The class also includes a *requires* attribute. This attribute is used for traceability of data through the graph. An example of its use would be if temperature data was passed to a transformer representing a Finite Element analysis used to calculate thermal expansion. The thermal expansion data instance, that was the output data of that transformer, would list temperature in its *required* attribute.

```python
class BaseEdge(TypedDict):
    id: str
    u: BaseNode
    v: BaseNode
    active: bool
```
(a)

```python
class DataEdge(TypedDict, BaseEdge):
    data: Data
```
(b)

```python
class Data(TypedDict):
    id: str
    name: Optional[str]
    description: Optional[str]
    frequency: Optional[str]
    size: Optional[str]
    requires: Optional[List[Data]]
```
(c)

Figure 3. Base edge definition (a), data edge definition (b), and data attribute definition (b)

3.4. Global rules

Beyond the definition of nodes and edges, there are several rules that are applied globally to ensure that the graph representation is valid. A valid graph means that all nodes obey the class definitions previously presented and that the following rules are adhered to:

1. **All data produced by a source must terminate at one or more stores or sinks, or be transformed within a transformer.**
2. **The graph must be acyclic. This means that a type of data must either be transformed, stored, or sunk; there can be no cycles.**

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press

Rule 1 ensures that all data produced by sources is used in some form or another. If a source produces data that is not utilised anywhere else in the graph, this data should not be included in the analysis. It is valid for data to be produced and terminate in a store, as this may be the case for incremental development of a DT, data might be collected now, and stored for future processing, but the data cannot be produced and go nowhere. Rule 2 necessitates an acyclic graph, this is to prevent infinite loops where a piece of data is perpetually moved in a cycle, without being utilised elsewhere. From a pragmatic standpoint, this constraint is essential for the stability of the cost algorithm presented in Section 4 as the presence of cycles would render the recursive calculations intractable.

4. Cost analysis at the conceptual design phase

This section details the proposed analysis methodologies that this schema will allow, before a vision demonstrator is presented in Section 5.

4.1. Path isolation

Sub-graphs can be generated that includes only the nodes and edges that directly contribute to a given sink node, enabling analysis of per-benefit cost calculations. The approach takes a directed graph of the entire DT, G and a target sink t as inputs. It then finds the set R of all ancestor nodes of t in the graph. An ancestor node is any node that has a directed path of any length to the target node. A sub-graph of ? is then created, which includes only the nodes in ?, as well as the target node t. The sub-graph includes all edges from G that connect the nodes in R. The pseudocode for this approach can be seen in Figure 4. The Benefit score associated with that sink is taken as the benefit side of the analysis, and the costs of the subset of the graph associated with that sink is compared against it. To consider multiple outputs simultaneously, the union of multiple subgraphs can be taken.

Algorithm 1 Get Ancestor Subgraph (Nodes that can reach t)

1: **function** GETANCESTORS(G, t) ▷ G is a MultiDiGraph, t is the target sink node.
2: $R \leftarrow$ ANCESTORS(G, t) ▷ Finds the set R of all nodes u in G such that a directed path $u \rightsquigarrow t$ exists.
3: $R \leftarrow R \cup \{t\}$ ▷ Ensure the target node t is included in the set.
4: **for** each node $u \in R$ **do**
5: SETATTRIBUTE(u, "active", True) ▷ Set an 'active' flag on all nodes in the ancestor set.
6: $G_R \leftarrow$ SUBGRAPH(G, R) ▷ Create a new graph G_R containing only the nodes in R and all edges from G that connect two nodes in R.
7: **return** G_R

Where:

- G: Represents the input directed multigraph.
- t: Represents the target "sink" node.
- R: Represents the set of ancestor nodes.
- $u \rightsquigarrow t$: This notation means "there exists a directed path from node u to node t".
- **Ancestors**(G, t): This procedural call maps to `nx.ancestors(G, target_sink)`.
- **Subgraph**(G, R): This maps to `G.subgraph(ancestors)`.

Figure 4. Algorithm to get subgraph for target node

4.2. Cost calculation

Each node in the graph has an associated cost attribute, which is defined as a function that returns a cost value for a given period. To establish the cost of a specific output, represented by a sink, the subset of the overall DT that contributes to that sink is identified using the path isolation algorithm. The cost calculation methodology traverses the path between nodes and calls the cost function of each visited node. This cost function increments a global cost counter. An overview of the algorithm to achieve this is shown in Figure 5.

By repeating this process for different graph configurations, designers can perform a comparative Cost Analysis between proposed options, allowing the design of the DT to be optimised before any physical implementation. A designer could also examine the impact of altering the parameters of elements of the DT, such as sensing rate or fidelity of a source, by calculating the downstream impact on the total cost of

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press

a specific sink node. Alternatively, when combined with the benefits assigned to each sink node, the cost-benefit trade-off of adding new elements to the DT could also be investigated, and the order of their implementation optimised to maximise the benefit accrued for each element added to the design.
To account for the uncertainty in early-stage design, probabilistic approaches could also be incorporated, to improve the accuracy of cost estimates. By assigning a probability distribution to each parameter, or to the output of the cost function, these distributions can be randomly sampled, forming a Monte-Carlo analysis to produce and overall probability distribution for the resultant cost.

Algorithm 2 Calculate cost of graph

```
1:  C_global ← 0                                  ▷ Initialize the global cost calculator.
2:  function CalculateGraphCost(G_R, t)           ▷ G_R is the ancestor subgraph, t is the sink node.
3:      V ← ∅                                     ▷ Initialize an empty set of visited nodes.
4:      G_T ← Transpose(G_R)                      ▷ Create a graph with all edge directions reversed.
5:      DfsVisit(G_T, t, V)                       ▷ Start the traversal from t on the reversed graph.
6:      return C_global                           ▷ Return the total accumulated cost.
7:
8:  procedure DfsVisit(G, u, V)                   ▷ G is the graph, u is the current node, V is the visited set.
9:      V ← V ∪ {u}                               ▷ Mark node u as visited.
10:     Cost(u)                                   ▷ Call Cost function for node u, which increments C_global.
11:     for each neighbor v of u in G do          ▷ Neighbors of u in the *transposed* graph.
12:         if v ∉ V then DfsVisit(G, v, V)
```

Where:

- C_{global}: Represents the global cost calculator, assumed to be accessible by all functions.
- V: Represents the set of nodes that have already been visited.
- $G_T \leftarrow$ **Transpose**(G_R): This operation creates a new graph G_T where every edge from G_R (e.g., $u \rightarrow v$) has its direction flipped ($u \leftarrow v$).
- **Cost**(u): A call to an external function that calculates the cost of node u and adds it to C_{global} as a side effect.
- **DfsVisit**(...): A recursive Depth-First Search procedure.

Figure 5. Cost calculation algorithm

5. Worked example

To demonstrate the schema in the design process for a Digital Twin, it will now be applied to a hypothetical case study that was produced using knowledge from the review of architecture during the development of the schema. The scenario for this case study is that a proposed DT design is being assessed for viability. The physical entity in this case study is a 3D printer, and the outputs are a predictor of the printer's next state, an email notification service to notify a technician that the print is complete, a report generator, and a delta between the bed temperature and the target bed temperature. Using the presented schema, this case study is represented as a graph in Figure 6. The labels on each edge represent the name of the data object associated with that edge.

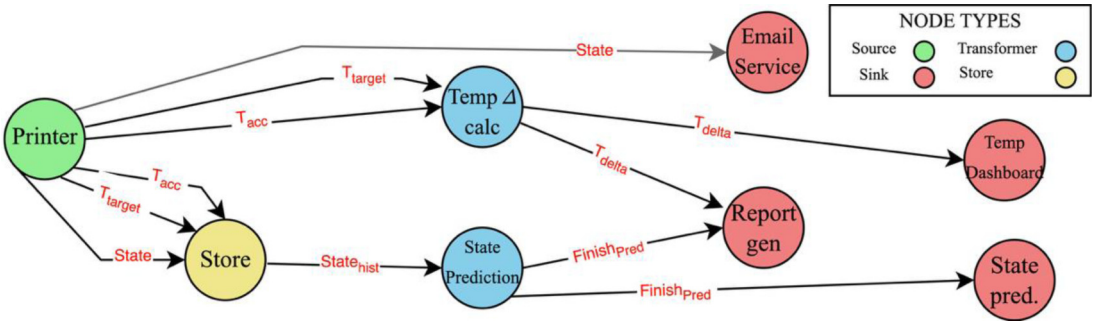


Figure 6. Case study architecture represented as a graph

To conduct the analysis, the graph was built using the networkx package for python, with nodes and edges created in accordance with the class definitions presented in Section 3. Each node contained

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press

attributes for each of the parameters outlined in its definition in Section 3. For the purposes of this example the cost function of each node was set as a constant, these are summarised in Table 2, along with illustrative benefits assigned to each output.

Table 2. Illustrative costs for nodes

Node Name	Printer	Store	Temp transformer	State Transformer	Email service	Temp dashboard	Report generation	State prediction
Cost (£/mo.)	10	5	3	4	1	2	4	2
Benefit	–	–	–	–	1	2	3	2

Using the path isolation algorithm detailed in Figure 4, the sub-graphs pertaining to the Report generation sink (RG), the State Prediction Sink (SP) and the Temp Dashboard Sink (TD) can be identified, and are shown in Figure 7 (a), (b), and (c) respectively.

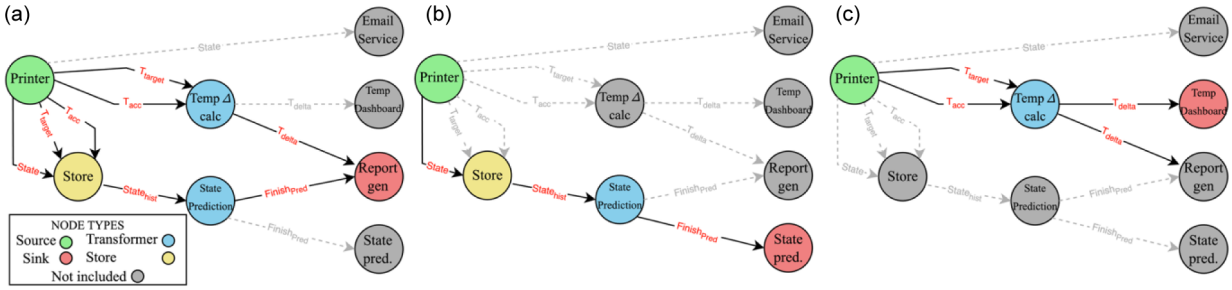


Figure 7. Sub-graph for each sink

The costs for each subgraph are calculated using the costing algorithm detailed in Figure 5, combined with the costs defined in Table 2, the results are shown in Table 3. For design options with multiple outputs, the cost estimate is compared to a bottom-up estimate, which involves aggregating the cost of each output individually.

Table 3. Comparison of costs of different outputs from the worked example

Output	RG	SP	TD	RG ∪ SP	SP ∪ TD	RG ∪ TD	RG ∪ TD ∪ SP
Cost (£)	26	21	15	28	26	28	30
Benefit	3	2	2	5	4	5	9
Cost per benefit (£)	8.7	11.5	7.5	5.6	6.5	5.6	3.3
Bottom-up estimate (£)	26	21	15	47	36	41	62
Cost difference (%)	0	0	0	51	32	38	70

The percentage differences between the estimate approaches demonstrates that traditional bottom-up approaches significantly overestimate the cost of implementation of these features when the estimate contains multiple outputs. This demonstrates that traditional methods fail to account for the interconnected nature of DTs and the benefits of using a graph-based approach. The cost per benefit score indicates the trade-off between cost and benefit in this case and can help to prioritise implementation to achieve an optimal implementation order. This analysis is summarised in Table 4, and shows that an optimal implementation order of $TD \rightarrow RG \rightarrow SP$ can be deduced.

Table 4. Optimal implementation order for worked example

Step	Action	System cost (£)	Marginal cost (£)	Marginal Benefit	Marg. Cost/marg. benefit
1	Add TD	15	15	2	7.5
2	Add RG	28	13	3	4.33
3	ADD SP	30	2	4	0.5

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press

6. Discussion and conclusions

The primary contribution of this work is the formalisation and extension of a DT architecture into a computable, algorithmically queryable topology. This representation through a formal graph schema provides a direct solution to the key challenges of DT cost analysis, particularly the problem of evaluating and attributing interconnected costs. As demonstrated through the worked example, bottom-up summations can provide overestimates of costs, as they fail to account for shared uses of nodes. It showed that bottom-up estimate, which aggregates the cost of individual outputs, overestimated the total cost by up to 70% for a system with three outputs (RG ∪ TD ∪ SP). This overestimation could be particularly prevalent in situations where siloed teams are developing components of the DT separately. The holistic graph representation allows opportunities for consolidation and therefore cost reduction to become apparent. The analysis framework provides better understanding of the marginal cost of adding new features and can help to identify areas where existing capabilities to be leveraged to achieve more benefits in the form of new outputs. Table 4 demonstrates that these capabilities can be utilised to determine an optimal implementation order, that maximises the cost-benefit proposition at each stage of the process.

Beyond costing, this form of representation can be used for overall systems analysis. The path isolation algorithm presented in Figure 4 can be used for impact analysis, looking at the propagation of failures if a node were to become inoperable, and identification of optimisation opportunities beyond pure cost optimisation.

There are several limitations to the current work, which in turn define the path for future research. The first limitation is that while the paper proposes a schema that can be used for full Cost-benefit analysis, it only provides a fully realised methodology for cost-estimation. The worked example includes indicative benefits, however, the methodology for quantifying these benefits and assigning them to specific business functions is an ongoing avenue of future work. To characterise the benefits and attribute them to Sink nodes, the future work will involve developing a database of benefits, and a ranking or weighting system to assign them as numerical values to the sink nodes, similar to the indicative values assigned in Section 5. Extensions to the schema to include dedicated nodes capturing these benefits are another possibility to be explored.

Another limitation is that the cost function attributed to each BaseNode, Figure 3(a), is currently presented as a node-specific black box. This paper proposed the framework and return type for these functions but did not define their specific structure. The practical implementation of this framework hinges on defining realistic cost functions for each node. This is another important area of future work and will involve empirical studies of real-world DT components to create a catalogue of standardised, parameter-driven cost functions. Another area of future work in this area is the implementation of probabilistic costing. This would be valuable as it would allow the appraisal of the potential range and uncertainty of the project.

The worked example utilised in this study is a relatively simple scenario. While useful for demonstrating the techniques presented, it does not accurately reflect the size and complexity of industrial DTs. Investigation of the schema's scalability and performance with highly complex DTs with large numbers of nodes, and intricate interdependencies is another important area of future work.

The schema presented in this work is designed to be used in the early stages of the design cycle, to determine an estimate of the cost-benefit proposition of potential DT projects. The paper has presented proposed analysis techniques to utilise the structure of the schema to calculate cost estimates for proposed architectures, and compare them to a metric quantifying the benefits. The analysis of the proposed architecture demonstrates the importance of considering costs as a union of services, rather than costs in isolation. The schema presented is sector agnostic, in that it can represent a DT for any purpose, and allows for iteration of multiple architecture concepts to decide on the optimal structure for a proposed Digital Twin.

References

Alonso, R., Locci, R., & Recupero, D. R. (2024). Improving digital twin experience through big data, IoT and social analysis: An architecture and a case study. *Heliyon*, 10(2), e24741. https://doi.org/10.1016/j.heliyon.2024.e24741

Brandes, U., Eiglsperger, M., Herman, I., Himsolt, M., & Marshall, M. S. (2002). GraphML Progress Report Structural Layer Proposal. In P. Mutzel, M. Jünger, & S. Leipert (Eds), *Graph Drawing* (Vol. 2265, pp. 501–512). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45848-4_59

Brealey, R. A. (2023). *Principles of corporate finance*. New York, Ny Mcgraw Hill Llc.

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press

Businesswire. (2023, March). Global Digital Twins Market Report 2023: Digital Twin Supported Solutions in Smart Cities Will Reach $5.2 Billion by 2028—ResearchAndMarkets.com. In *Businesswire*. https://www.businesswire.com/news/home/20230320005400/en/Global-Digital-Twins-Market-Report-2023-Digital-Twin-Supported-Solutions-in-Smart-Cities-Will-Reach-%245.2-Billion-by-2028—ResearchAndMarkets.com

Deo, N. (1974). *Graph Theory with Applications to Engineering and Computer Science*. PHI Learning Pvt. Ltd.

Diestel, R. (2018). *Graph theory*. Springer.

Grieves, M. (2015). *Digital Twin: Manufacturing Excellence through Virtual Factory Replication.*

Guo, W., Tian, Q., Jiang, Z., & Wang, H. (2018). A graph-based cost model for supply chain reconfiguration. *Journal of Manufacturing Systems*, 48, 55–63. https://doi.org/10.1016/j.jmsy.2018.04.015

Gürdür Broo, D., Bravo-Haro, M., & Schooling, J. (2022). Design and implementation of a smart infrastructure digital twin. *Automation in Construction*, 136, 104171. https://doi.org/10.1016/j.autcon.2022.104171

Heckmann, T., Schwanghart, W., & Phillips, J. (2014). Graph theory—Recent developments of its application in geomorphology. *Geomorphology*, 243. https://doi.org/10.1016/j.geomorph.2014.12.024

Hexagon. (2024). 2024 Digital Twin Industry Report Hexagon. In *Calameo.com.* https://www.calameo.com/read/001915796155d0c494c32?authid=L8cWi6laffjB

Himsolt, M. (2010). *GML: A portable Graph File Format.*

Insights, F. B. (2025). Digital Twin Market Size, Share, Growth | Analysis [2021-2028]. In *Www.fortunebusinessinsights.com*. https://www.fortunebusinessinsights.com/digital-twin-market-106246

International Organization for Standardization, International Electrotechnical Commission, & Institute of Electrical and Electronics Engineers. (2023). *Systems and software engineering – System life cycle processes* (ISO/IEC/IEEE 15288:2023). International Organization for Standardization. https://www.iso.org/standard/81702.html

Khan, A. Q., Matskin, M., Prodan, R., Bussler, C., Roman, D., & Soylu, A. (2024). Cost modelling and optimisation for cloud: A graph-based approach. *Journal of Cloud Computing*, 13(1), 147. https://doi.org/10.1186/s13677-024-00709-6

Lengthorn, P. (n.d.). Cost to build a Digital Twin. In *Www.consultengsurvivor.com*. Retrieved https://www.consultengsurvivor.com/cost-to-build-a-digital-twin

McClenaghan, A., Gopsill, J., Ballantyne, R., & Hicks, B. (2023). Cost Benefit Analysis for Digital Twin Model Selection at the Time of Investment. *Procedia Cirp*, *Vol* 108.

Mckinsey. (2022, October). Digital twins: From one twin to the enterprise metaverse | McKinsey. In *Www.mckinsey.com*. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/digital-twins-from-one-twin-to-the-enterprise-metaverse

Mihai, S., Yaqoob, M., Hung, D. V., Davis, W., Towakel, P., Raza, M., Karamanoglu, M., Barn, B., Shetve, D., Prasad, R. V., Venkataraman, H., Trestian, R., & Nguyen, H. X. (2022). Digital Twins: A Survey on Enabling Technologies, Challenges, Trends and Future Prospects. *IEEE Communications Surveys & Tutorials*, 24(4), 2255–2291. https://doi.org/10.1109/COMST.2022.3208773

NASA. (2023, June). NASA Cost Estimating Handbook (CEH)—NASA. In *NASA*. https://www.nasa.gov/ocfo/ppc-corner/nasa-cost-estimating-handbook-ceh/

Patzwahl, F., & Hyba, M. (2022). *An Approach for System Analysis with MBSE and Graph Data Engineering*. https://doi.org/10.48550/arXiv.2201.06363

Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M., & Vrgoč, D. (2016). Foundations of JSON schema. *Proceedings of the 25th International Conference on World Wide Web*, 263–273.

RisingMax. (2024, January). How Much Does It Cost To Develop A Digital Twin Application. In *Risingmax.com.* https://risingmax.com/blog/digital-twin-development-cost

Thomas, D. S. (2024). *Economics of Digital Twins:* https://doi.org/10.6028/nist.ams.100-61

https://doi.org/10.1017/pds.2026.10429 Published online by Cambridge University Press