Smart Solutions for Identifying Compatible Components – Powered by metaphactory and RDFox

A Configuration Management Use Case

www.metaphacts.com
oxfordsemantic.tech

metaphactory

RDFOX

Table of contents

1. Identifying compatible components

Determining compatibility between individual entities is an essential process for many businesses, across various industries and business models; from industrial configuration, supply chain, bill of materials, evaluating terms in contracts, or even for match making apps.

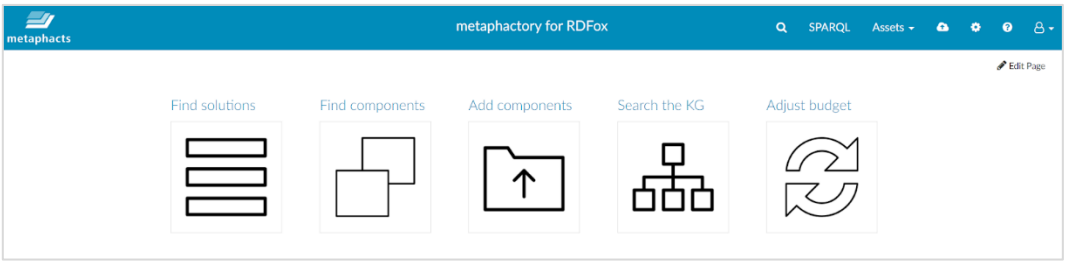
The process may sometimes require the user to check hundreds of thousands or millions of possible combinations, to assess whether components fit together, or if components meet specified requirements. Additional factors may also need to be taken into account, for example, regulations or customer budgets.

Traditional approaches are inefficient for modern day applications due to the large volumes of data, heterogeneity of data formats, complexity of customer specifications, and concerns over scalability.

In industrial configuration scenarios, for example, users need to identify suitable solutions by looking at hundreds or thousands of technical parts, each with their own set of attributes and constraints. In this case, the process of assessing compatibility can take up to 15 hours and involve numerous data export and aggregation steps as well as cross-checks across multiple systems - and this just for *one* compatibility solution. This process can be error-prone and is inefficient when businesses are meeting real-time customer demands. Further challenges may arise if the system changes; in relational systems, for example, the whole database must reload, which is a timely process, and checks must be performed to ensure that the changes haven't impacted the correctness or consistency of the application.

These concerns can be overcome through the use of our innovative **knowledge graph-based application** built on top of **metaphactory**, a knowledge graph management, visualisation and interaction platform, and **RDFox**, a knowledge graph and semantic reasoning engine. Knowledge graphs have structural advantages over traditional models for solving compatibility solutions. They overcome the flexibility limitations of relational databases as data is stored as richly connected entities, the system can be updated easily, and end-users can undertake targeted search, discovery and exploration. Knowledge graphs ensure data is **FAIR** - Findable, Accessible, Interoperable and Reusable.

This article will outline the metaphactory and RDFox joint solution for compatibility assessment, using an industrial configuration use case example, to demonstrate the unrivalled performance it gives end users.



2. metaphactory and RDFox in action for configuration management

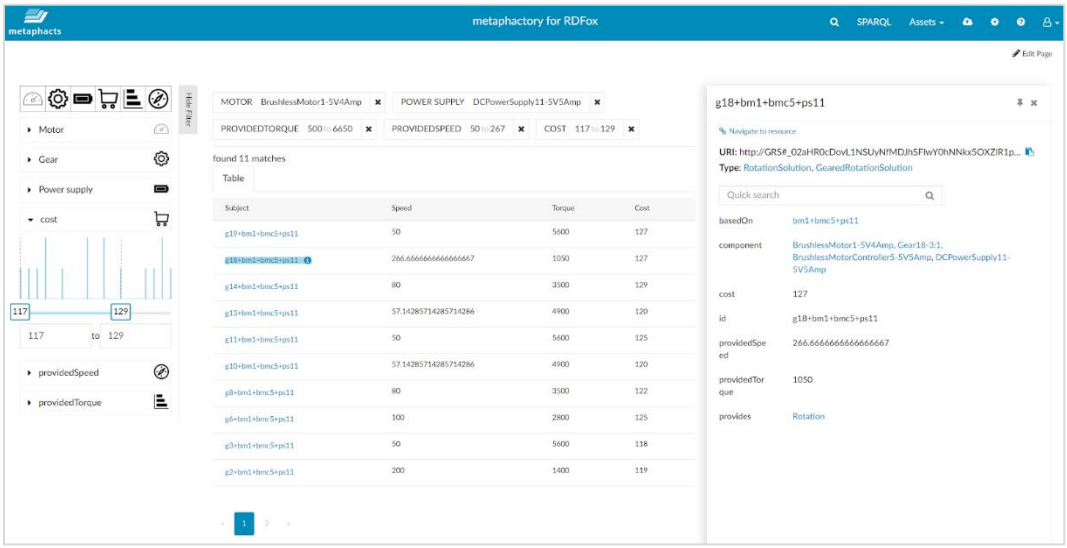
Together, metaphactory and RDFox deliver unprecedented results in compatibility determination scenarios by allowing users to quickly and efficiently gain access to actionable and meaningful insights.

The following section explores a knowledge graph-driven application for industrial configuration management where support engineers, product managers, technical planners, or technical maintenance specialists can quickly evaluate hundreds of components such as motors, gears, switches, power supplies and controllers, and their individual characteristics. **The user can determine how components fit together and how they can be combined to create solutions that solve very specific customer needs or maintenance requests, while staying within a predefined budget.**

2.1. Compatible rotation solutions at your fingertips

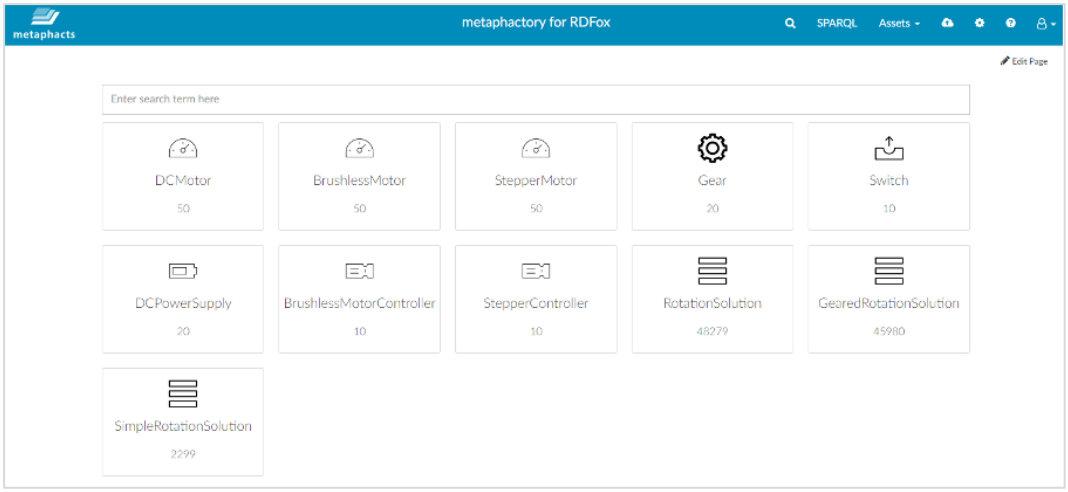
Let's consider the example of a support engineer who is looking for a complete rotation solution for one of her customers. The solution should come with a particular brushless motor, a specific power supply, a minimum speed (50 rpm) and torque (500 Nm), but should stay within a certain budget (max. 129€).

Compatibility between the components represented in the knowledge graph is determined by reasoning (Datalog rules) applied by RDFox. RDFox also automatically computes the cost of each solution based on the individual costs of the components. Using an intuitive interface built with metaphactory, the engineer can explore available solutions by simply selecting the components that should be included and defining the constraints that should be respected.

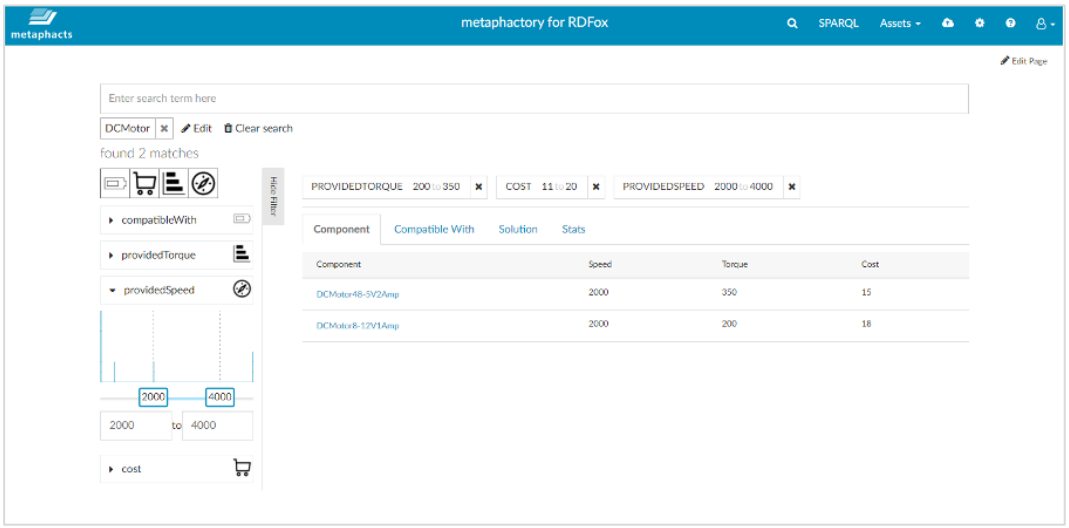


Once she has found a few solutions that satisfy her needs, the engineer can proceed to look at each one in detail and decide on the one she wants to order for her customer.

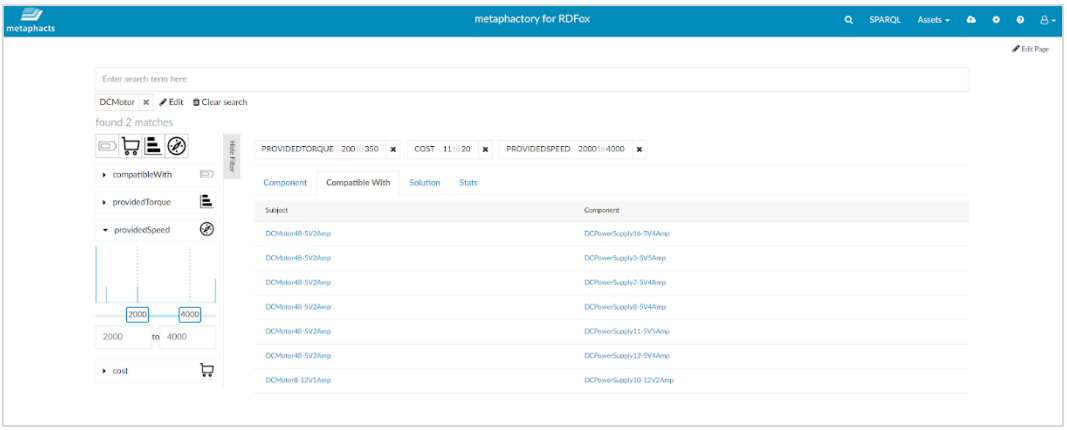
Now let's say our support engineer is assigned to replace a faulty DC motor at a customer site, but all other components that are part of the customer's existing rotation solution should be kept. Using the configuration management application built with metaphactory and RDFox, she can start by looking at the various DC motors available and filter down to find the ones compatible with the technical setup the customer has in place.



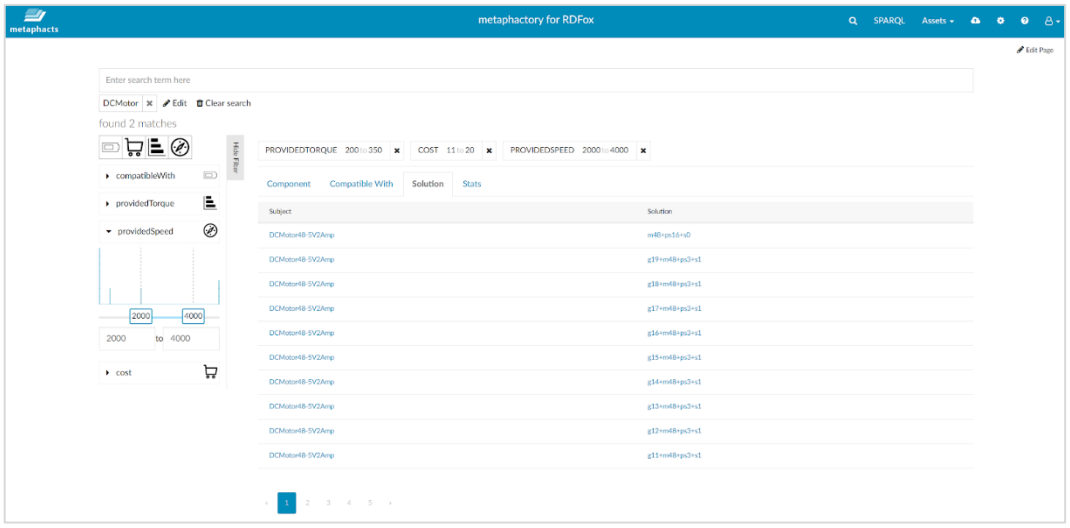
For example, she might be looking for a DC motor with a minimum torque of 200 Nm and a provided speed between 2,000 and 4,000 rpm, but which should not go over a predefined budget of €20.



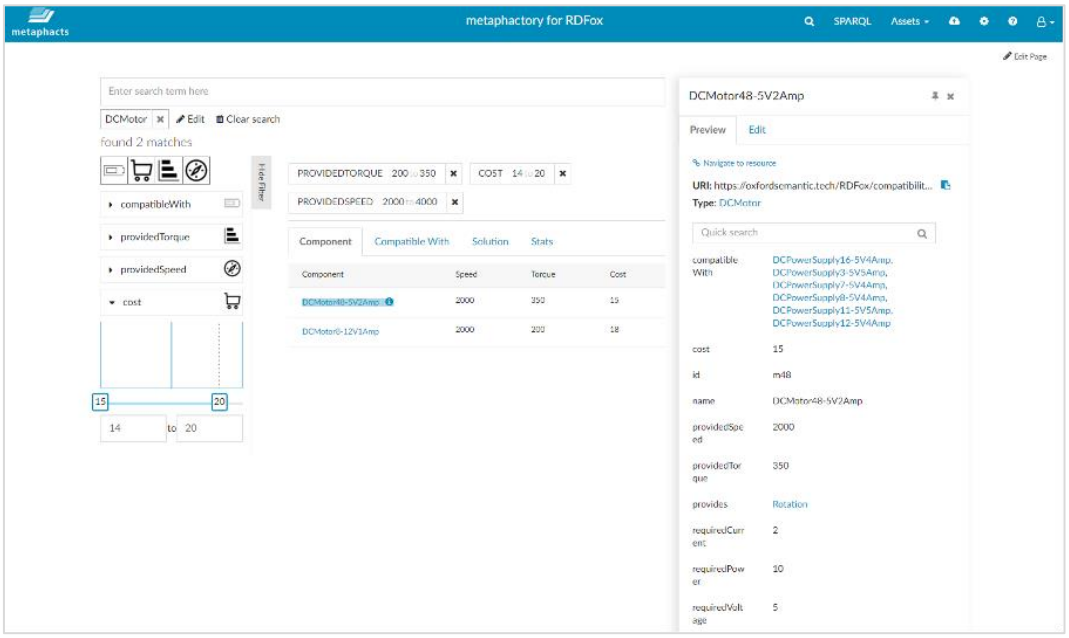
After a quick search based on her parameters, our engineer can go on to further explore her search results, for example which other components the resulting DC motors are compatible with:



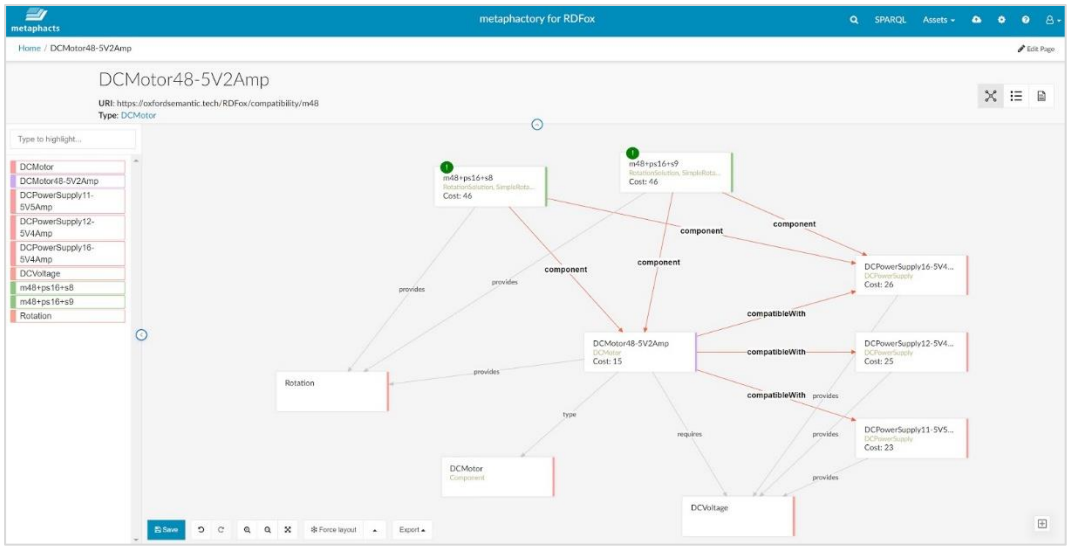
and which rotation solutions these DC motors are part of:



She can also look at each DC motor's specifications in detail:



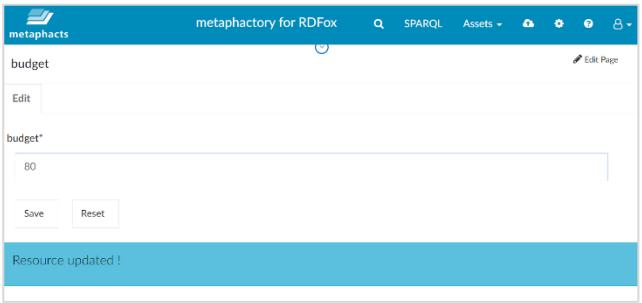
or visually explore relationships between a DC motor and other components:



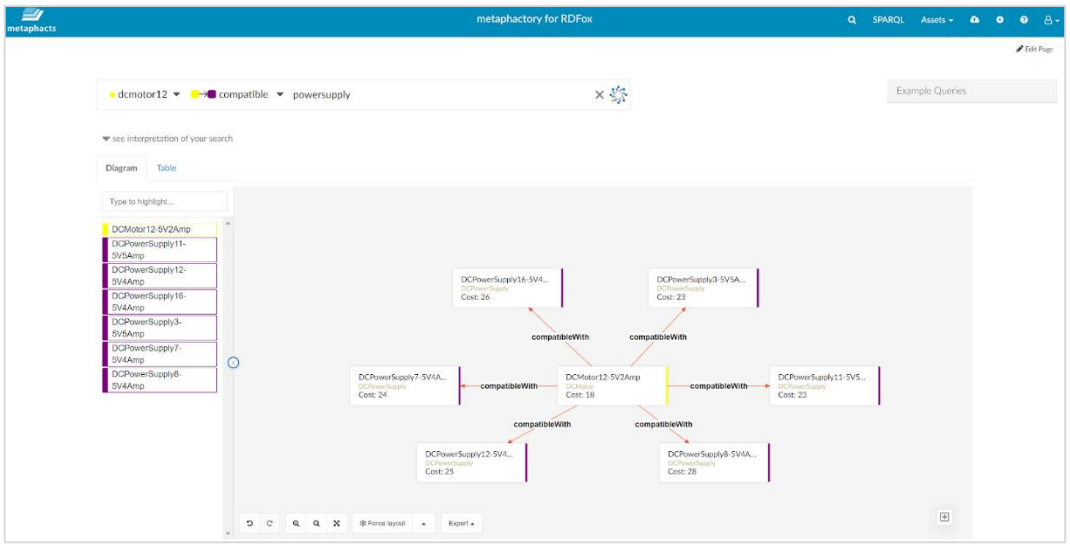
Using the visual exploration component integrated into the application, our end user can quickly build a diagram to show power supplies this DC motor is compatible with, as well as rotation solutions it is part of. The engineer can also find out at a glance which information was initially loaded into RDFox (e.g., the grey "type" relationship tells us that this information is core information) and which information was inferred based on the rules defined with RDFox (e.g., the red "compatibleWith" or "component" relationships tell us that this is inferred information).

2.2. Precise answers to specific questions

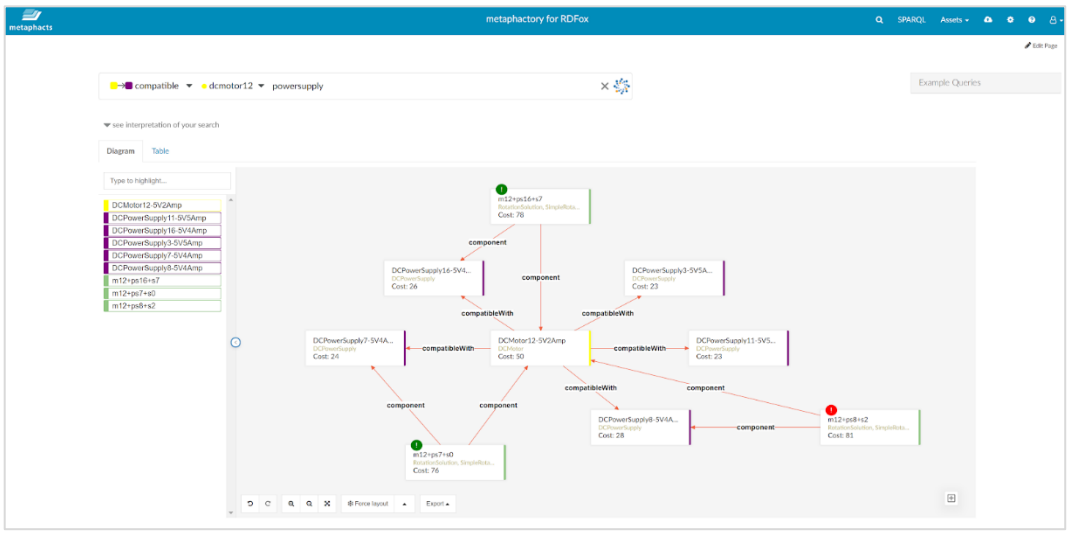
With its keyword interpretation engine and its intuitive visual graph exploration component, metaphactory allows end users to leverage core and inferred graph data in RDFox and perform targeted, natural-language queries that deliver instant results and can be explored further to discover previously unknown connections. In the example below, our engineer is searching power supplies compatible with a particular DC motor (DCMotor12) to set up a rotation solution for a customer. She starts by defining a customer budget of €80 for the complete solution.



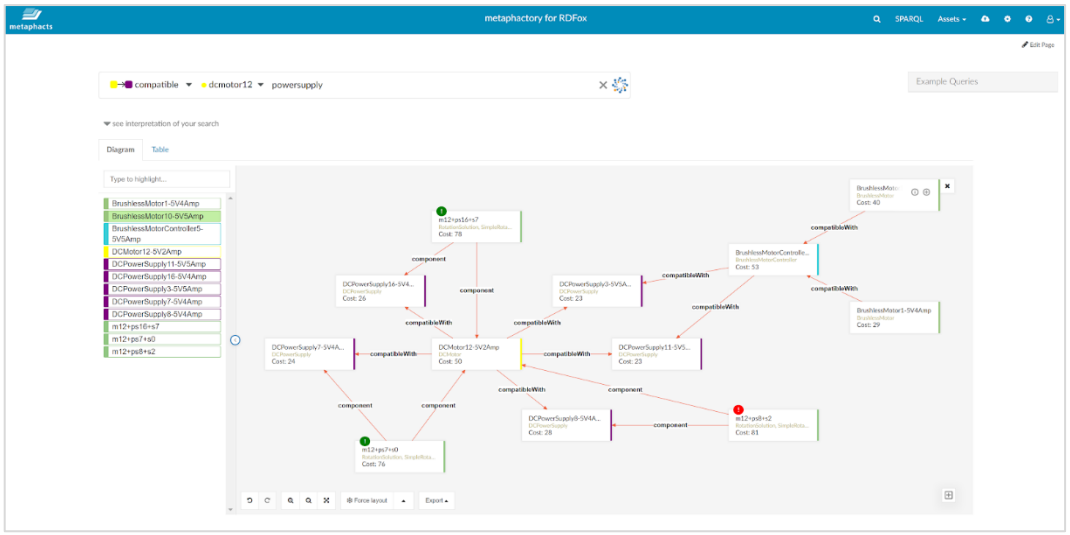
Then, she searches for all power supplies compatible with DCMotor12. She can type her keywords in the search bar and the system immediately returns a visual graph with all matching components. Note how metaphactory's keyword interpretation engine understood that the keyword "powersupply" refers to the "DCPowerSupply" entities stored in the knowledge graph.



She then explores the data in the system further to find rotation solutions that include these components. Note that two rotation solutions including two different power supplies are within budget, while a third one slightly exceeds the budget.



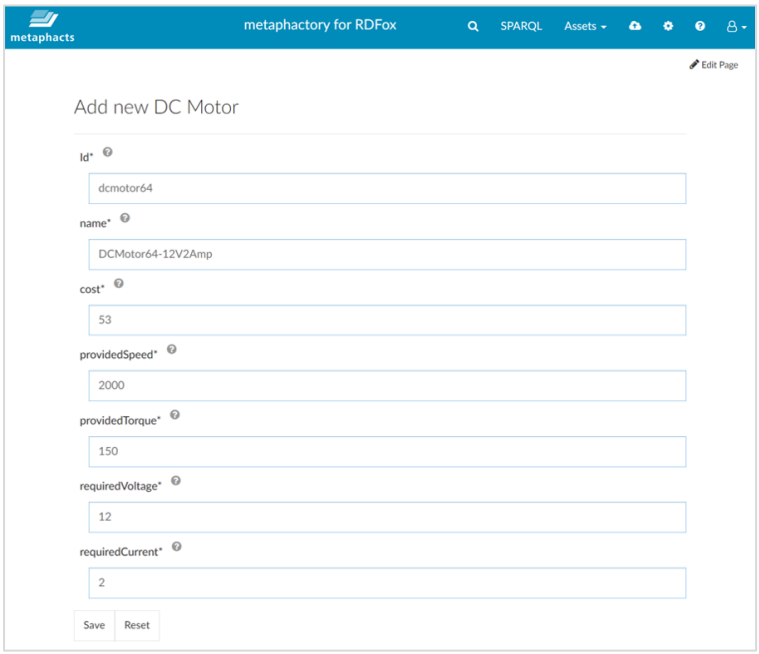
From here, our engineer can further explore relations in the graph and discover that two of the power supplies listed are also compatible with a brushless motor controller, which in turn is compatible with multiple brushless motors.



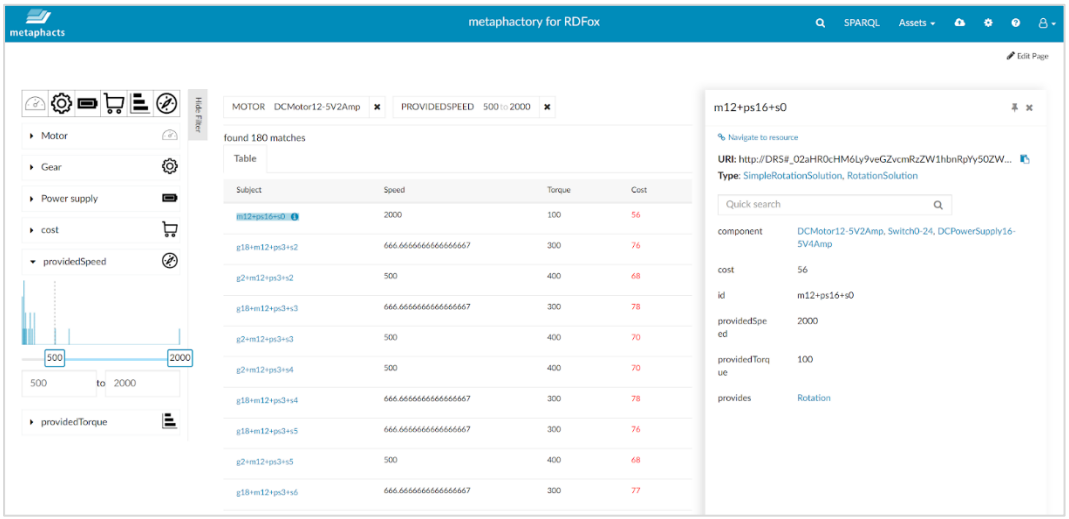
2.3. Immediate access to the latest data

As is explained in the following section, using RDFox, knowledge graph experts can easily import new data along with a set of compatibility rules and the system will automatically reflect the changes in the data, thus always giving end users access to the latest data. Similarly, data can be deleted or the ontology can be adjusted to reflect changes in relationships, and the system and the user experience created with metaphactory will update accordingly.

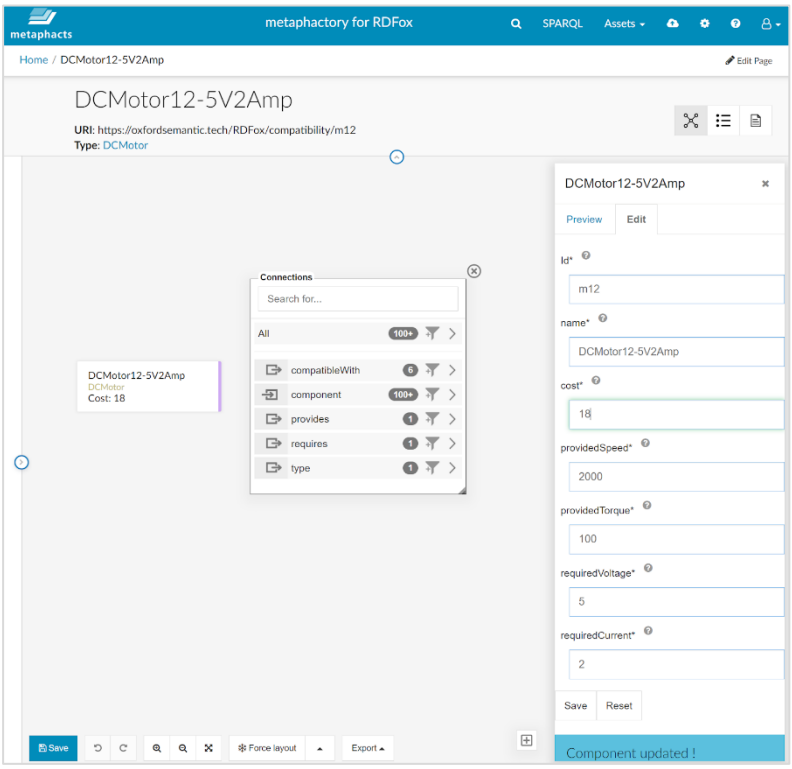
Often there is also a need for end users to be able to modify or augment the data in the knowledge graph. Using metaphactory's intuitive semantic forms, end users are able to seamlessly add new components to their catalogue:



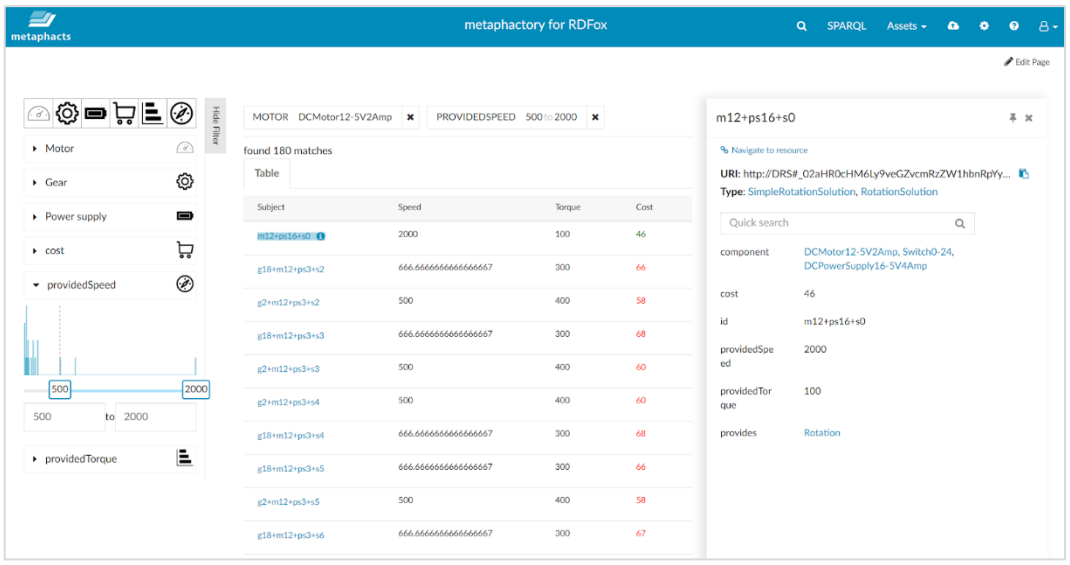
Similarly, end users can change the configuration of existing components and the updates will immediately be propagated throughout the entire user experience. Let's say, for example, that our support engineer from before is again looking for a rotation solution with specific parameters but within a predefined budget of €50. As depicted by the screenshot below, all of the rotation solutions in the system exceed the €50 budget.



After some negotiations with her supplier of DC motors, our engineer can adjust the price for the DC motor that should be included in the customer solution:



This will result in multiple rotation solutions being updated accordingly:



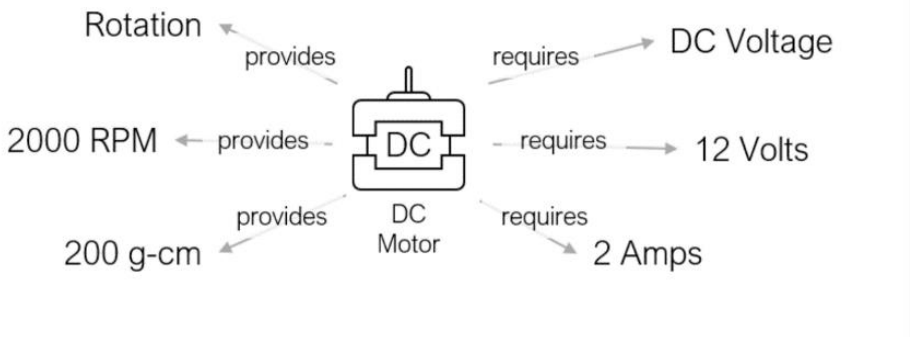
3. Behind the scenes: Achieving efficient and intuitive compatibility assessment solutions with RDFox and metaphactory

3.1. Laying the groundwork: Creating a Knowledge Graph with RDFox

RDFox, an in-memory RDF triple store for building knowledge graphs, can be used to make compatibility solutions correct, with almost unnoticeable iteration times, and operate 50–100 times faster than other RDF stores. It offers a unique proposition through being an in-memory solution and its advanced reasoning capabilities allow compatibility logic to be encoded, data to be added and removed, and new facts materialised.

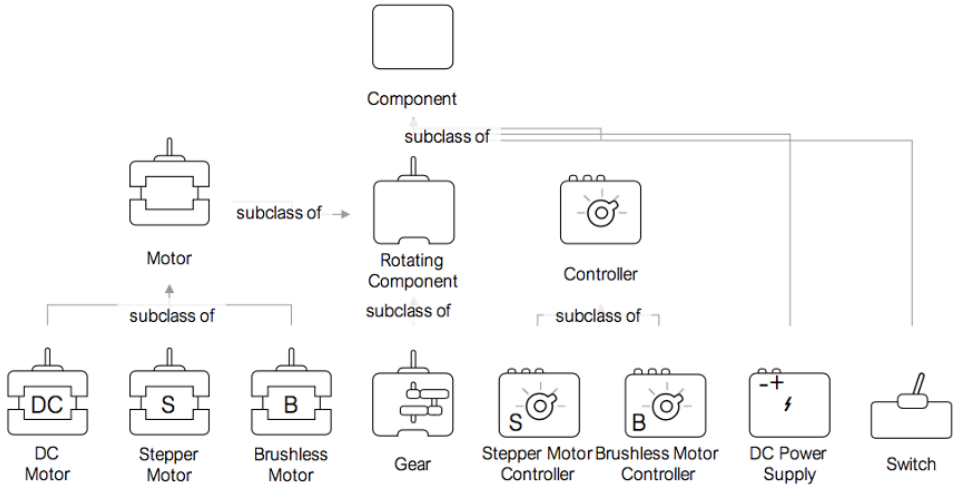
3.1.1. Attributes and hierarchies

To begin the process, each component's attributes, including its provisions and requirements, are mapped into the knowledge graph. For this example, there are 500 components each with a specific set of provisions and requirements, which must be included so that compatibility can be assessed.



Example of DC Motor, its provisions and requirements

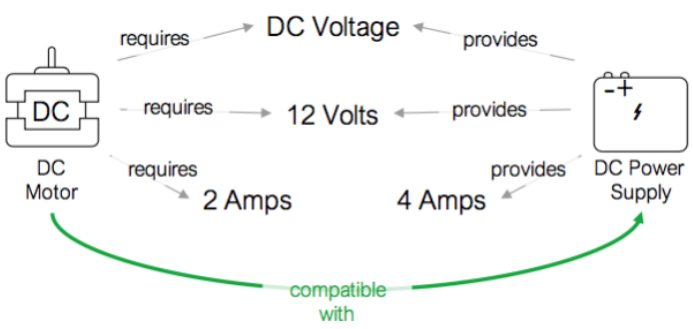
To capture and embed component hierarchies and organize the domain-specific knowledge, we can add an ontology (or a data model) into the graph. This provides structures and relationships which helps with querying the dataset and writing rules.



3.1.2. Rules and reasoning

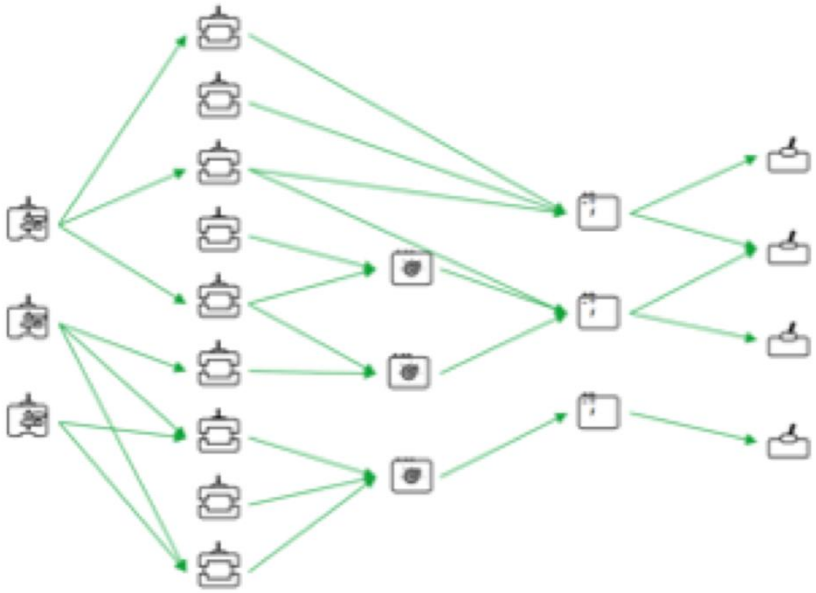
Once the components are stored within the knowledge graph, reasoning is used to determine the compatibility of components. In this context reasoning is the ability to understand and apply the logic of component compatibility, easily and correctly into the system, in order to meet customer's requirements. This is done using rules expressed in Datalog. A comparison of the constraints and provisions of components is carried out and compatible configuration solutions are stored in the knowledge graph as new, inferred relationships between components.

The following rule for compatibility demonstrates that rules are expressions of logic, representing an 'if… then' statement. This rule says: *if the power supply provides a DC Voltage, a voltage of a specific amount, and a current of a specific amount, and the motor requires a DC Voltage, a voltage of the same amount, and a current of the same amount, then the two are compatible.*




```
[?M, :compatibleWith, ?PS] :-
  :DCMotor[?M] ,
  :DCPowerSupply[?PS] ,
  [?PS, :provides, :DCVoltage] ,
  [?PS, :providedVoltage, ?pv] ,
  [?PS, :providedCurrent, ?pc] ,
  [?M, :requires, :DCVoltage] ,
  [?M, :requiredVoltage, ?rv] ,
  [?M, :requiredCurrent, ?rc] ,
  FILTER (?pv = ?rv && ?pc >= ?rc) .
```

There are numerous chains of compatibility throughout the dataset which are calculated and stored in the graph. This allows for various pathway options for configuration solutions. As these are calculated ahead of query time and stored in the graph, query performance is significantly improved compared to other graph databases or SQL solutions, where compatibility is calculated during the query process.



3.1.3. Changing the system

Modern applications require flexibility, for example, components may be added or removed from the database or constraints might change. Businesses which use traditional methods with long iteration periods suffer in this regard. With RDFox, new component data can be imported, along with new rules for compatibility at any point. Additionally, because everything is expressed using RDF, the ontology can be flexibly extended or adjusted at any time to account for changes in the business use case, component characteristics or component hierarchies.

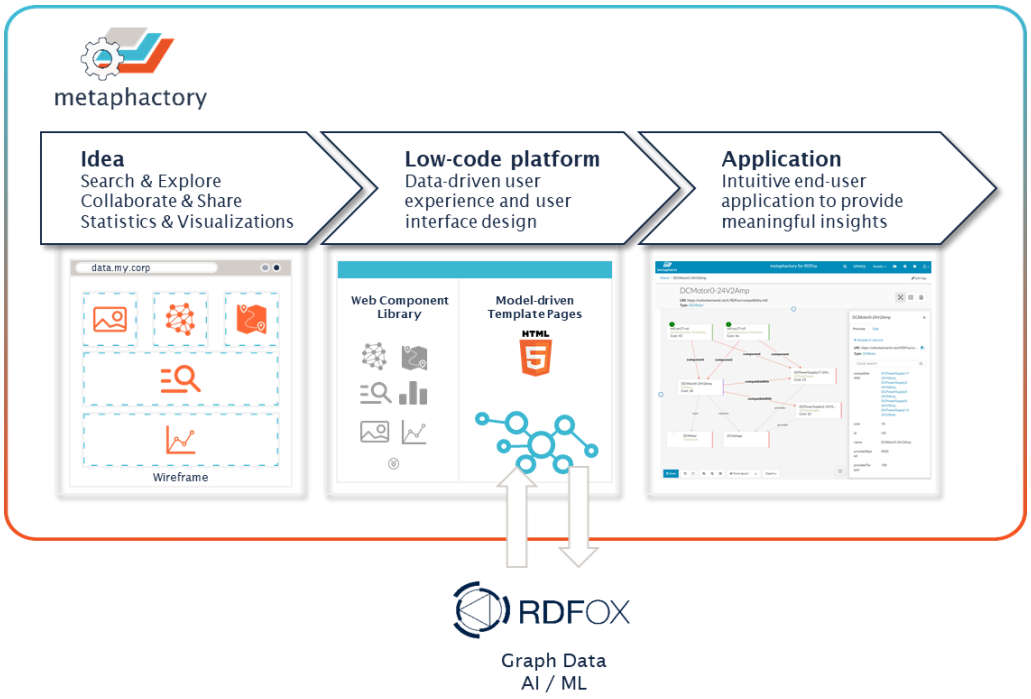
The incremental capabilities allow this to be achieved without needing to update the full system. Similarly, removing components requires a simple delete command, which will also remove all relationships attributed to that component incrementally.

Constraints often change due to budgets, environment, demand, and so on. Rules can be updated to reflect these changes incrementally and with almost negligible iteration times. As such, RDFox provides flexibility without compromising on performance.

3.2. Building an intuitive user experience for interacting with the Knowledge Graph using metaphactory

metaphactory is a FAIR Data, Knowledge Graph platform which supports customers in unlocking the power of their data. It runs on top of the RDFox graph database and delivers capabilities and features for knowledge graph management, rapid application development, and end-user oriented interaction. Its generic approach based on open standards offers great flexibility in different use case scenarios and across various industries and application areas.

As a low-code platform, metaphactory can be used to rapidly build data-driven, end-user facing applications on top of the knowledge graph and allows customers to go from idea to production within a few weeks.

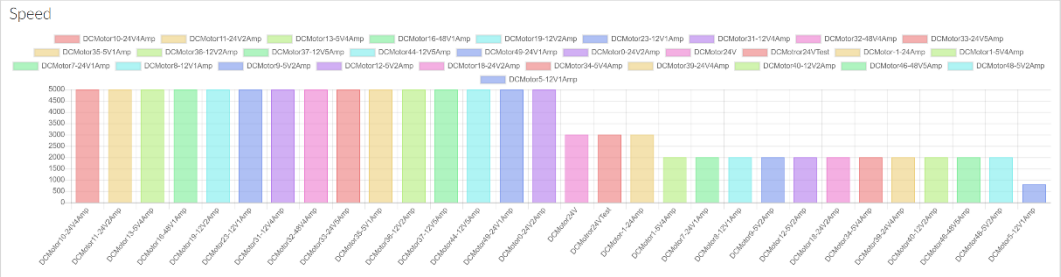


Low-code platform approach to go from idea to production in a few weeks

Using a powerful templating engine, knowledge graph instances pertaining to the same class/type can be rendered in a unified way. Template pages are populated by semantic Web components that can be parameterized to meet specific information needs. For example, in the industrial configuration scenario, the template page for a DC motor might display a knowledge panel with the motor's specifications, a visual graph interaction component to discover relations to other components, a table showing all compatible components which can be filtered by type, and an edit form to modify the characteristics of the component. The data populating these components is pulled directly from the knowledge

graph using SPARQL queries pre-defined by the application developers.

The rich set of semantic Web components can additionally be used to build custom dashboards for search, visualization, interactive exploration and authoring.



Distribution of DC motors based on speed

This empowers end users and allows them to consume data and extract meaningful, actionable insights without having to understand complex data models, write technical queries or go through tedious processes.

The result is a rich and intuitive end user experience that can be flexibly adjusted, extended, enhanced and reused to meet changing business needs and requirements.

4. The bottom line

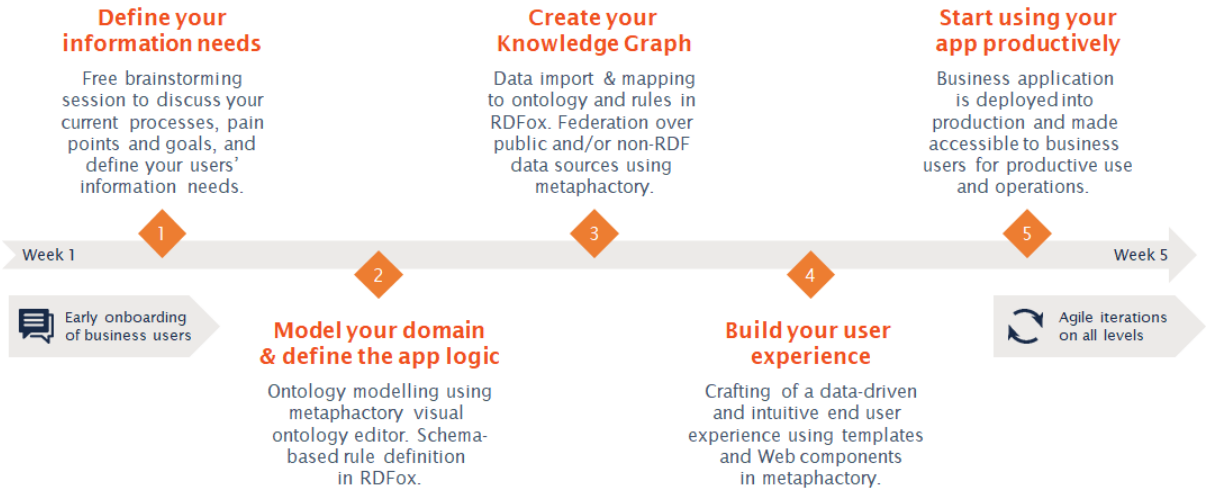
The metaphactory-RDFox joint solution offers a smart, unique and flexible method for determining compatibility solutions. With ontologies and semantic reasoning, the integration of hierarchies and logic brings the intelligence layer closer to the data. The metaphactory platform enables the end user to extract intelligible insights from the complex data and to explore this data through custom-built dashboards, optimised for search, visualisation, interactive exploration and authoring. For the user, the result is an adaptable solution, which can react to real-time changes.

Incremental reasoning capabilities allow the addition or removal of information and the adjustment of interaction patterns with almost negligible iteration periods, a feature which is far more appropriate for modern-day applications than slower, traditional methods. The usability and speed of the joint solution is bolstered by the ability to determine compatibility solutions ahead of query time, storing the new solutions as relationships within the knowledge graph, resulting in improved query performance and user satisfaction.

The end-to-end solution provides the chance for organisations to optimise their compatibility process with advanced technologies, customised with their business domain knowledge and expertise.

5. Get started with metaphactory and RDFox

The illustration below outlines the anatomy of a standard customer project based on metaphactory and RDFox. We guide and support customers through the entire process of building a FAIR data, knowledge graph-driven application that fits their needs. This covers core steps such as identifying main use cases, domain modelling and defining the application logic, knowledge graph creation, crafting the user experience and user interface, and deployment into production. All of this happens in a very agile manner and in close collaboration and synchronisation with your team of experts and selected end users, who have the opportunity to provide feedback and help refine the end solution at every step along the way. Typically, the implementation of a joint metaphactory-RDFox project takes approx. 5 weeks, depending on the use case scope and complexity - an unimaginable timeframe for traditional solutions!



metaphacts

Oxford Semantic Technologies

Get started: metaphacts.com/get-started

Contact us: sales@metaphacts.com