>How has compatibility with RDF/OWL benefited SysMLv2?

Aligning SysML v2 -- and in particular the underlying Kernel Modeling Language (KerML) -- with RDF/OWL2 is done because RDF / OWL2 / Semantic Web is the most mature and most widely used implementation of first order logic. This provides a strong logic / mathematical semantic basis that was carefully designed to provide the best balance between expressivity and decidability, in other words: it allows automated reasoning algorithms to complete the computation of the semantic consequences of a model in finite time.

SysML v2 / KerML is not yet benefitting from RDF/OWL2 and also there is not yet an executable transformation from a SysML v2 model to an RDF/OWL2 representation, so currently compatibility between SysML v2 and RDF/OWL2 cannot be claimed. However, it is expected to become available, some groups are working on this. One important difference is that without further constraints an RDF/OWL2 ontology follows the "Open World" assumption while a SysML v2 model follows a "Closed World" assumption, see https://en.wikipedia.org/wiki/Open-world_assumption for a detailed explanation.

So, all in all the approach is to provide a solid first order logic foundation as from experience this is a necessary condition for reaching semantic interoperability.

> In which cases a mix of diagrams and text were found to be most useful?

Textual and graphical notation have both their advantages and disadvantages: e.g.:

- text allows to be more precise, use computable expressions, and fits easily with version / configuration control / detecting differences,

  but cause to not "see the forest for the trees". Modern text editors help with automated folding etc.

- graphical provides a better overview, shows the bigger picture and how elements are connected, in particular for architectural views with filtering / layering / auto-layout provided by MBSE tooling, however it can be difficult to be precise or be sure that what you see in a diagram reflects exactly what is in the model.

SysMLv2 strives to achieve full compatibility between textual and graphical notation with bi-directional transformation. However, this is intrinsically hard, so it will not be 100% in practice. The idea is that the SysML v2 specification does not impose one or the other, but that users can decide and optimise what mix of textual and graphical notation works best for their application. In addition there is a strong Viewpoint and View definition capability built into the standard, that allows to predefine any suitable mix of textual, tabular, tree, interconnection, decomposition, and user-defined viewpoints & views, and specify how they should be rendered.

>Is sysmlv2 already supported by simulation or other analysis tools to validate the models and the defined semantics?

SysML v2 comprises capabilities to specify paratemerised Analysis Cases and Verification Cases, and bring back results into the SysML model. Also meta-data to allow connection to external analysis solvers, simulation tools etc. As part of the validation e.g. Ansys Phoenics Integration are prototyping interfaces and validation cases with their Model Center tool, and also others experiments with Matlab and Modelica based simulators are in progress. In addition there is associated research on making SysML v2 execution semantics and so-called "legal execution traces" precise and consistent. Smaller analyses / simulations could be executed inside the SysML model itself, as the textual language is rich enough to define algorithms. There is a library to support state-space representation models. There is also support for synchronous and asynchronous / discrete event or continous simulation. Finally traditional Use Cases are now modelled using the same constructs/pattern, so in principle would be amenable to early analysis / simulation.

>Should a company invest now in SysML or wait for v2 ?

It depends ;-) This is difficult to answer in general. Clearly, SysML v2 is not yet ready for production use for a while. The full standard is scheduled to be available by Feb 2022. As stated in the presentation the best answer we have from major MBSE tool vendors (both SysML v1 and other) is that it will take 2 ~ 3 years to implement the standard for real industrial use, so by March 2024 earliest. On the other hand, the existing prototype implementations (checkout https://github.com/Systems-Modeling) is already quite sophisticated and can actually be used in smaller / experimental settings, also to influence the use of SysML v1. How quickly the tool developers will implement SysML v2 -- or make non-SysML tools interoperable with the SysML v2 API -- will depend also on how much pressure the end-user community (including big organisations like Airbus, Thales, Boeing, Lockheed Martin, ESA, NASA, etc.) is able to exert.

So, it depends on circumstances of the particular organisation what is wise or not. However both for organisations that are just considering or starting MBSE and for those that already use it in daily practice, I think it is a good idea to take a very serious look at SysML v2, because a lot of wide industrial experience, lessons learnt, and new ideas by an excellent team of early MBSE adopters went into the development of SysML v2 (the specifications, the training material and the pilot tooling). Since it is all freely available from github under open licences the only investment cost is engineer hours from the organisation. If nothing else, it will help to sharpen / evolve / complete the definition of the organisation's MBSE strategy.

>What kind of tools / technology is used to develop the REST api to the model?

The HTTP/REST API is specified using the OpenAPI specification version 2.0, see https://spec.openapis.org and https://swagger.io/specification/v2/ .

There is a public dynamic documentation and test server available at http://sysml2-sst.intercax.com:9000/docs/

The underlying software is the Play Framework in Java (https://www.playframework.com/)

The OSLC API is also specified using the OpenAPI specification version 2.0 and implemented by IBM using their open source OSLC libraries.

For full specification details refer to https://github.com/Systems-Modeling/SysML-v2-Release/blob/master/doc/3-Systems_Modeling_API_and_Services.pdf clauses 8.1 and 8.2.

>Are there ways to formally define subsets of SysML2? Are there any such things already defined?

Yes. The specification has so-called conformance classes, that defines coherent subsets of functionality. They are supported by conformance test suites to permit verification.

Currently only part 3 on API and Services has a specified conformance test suite. For the SysML v2 spec it is scheduled for the final submission in Feb 2022.

>One of the challenges in exchanging information in SysML v1 using XMI are the toolspecific extentions, how is this addressed in SysML v2

SysML v2 does not make use anymore of the rather informal UML profile with stereotypes approach to language extension. The KerML and SysML metamodels are kept as simple as possible, and supplemented by normative, semantic libraries at user-model level (OMG level M1). Language extension is done by extending these normative libraries with specializations that provide the desired domain specific capabilities. There is support for user-defined keywords and metadata in the textual language.

As a result, extensions are much better controlled, than was the case with SysML v1. Also the new API and Services allows for much more fine-grained and dynamic access to a SysML v2 model repository, which should also allow for better and more practical access to or repression of tool-specific extensions. Furthermore there is a much more developed Viewpoint and View capability that allows to define in a tool-independent way views of the model information to be viewed and how it should be rendered. This will take care of many use cases for which tool-specific extensions are used today.

However as with any extension beyond the scope of the standard, a receiving tool will need to implement some kind of processing to handle extensions.