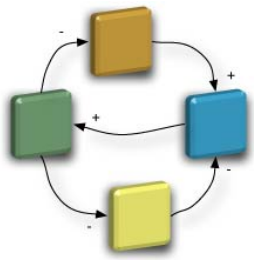
TRAK - AN ARCHITECTURE FRAMEWORK FOR RAIL

Case Study: An Implementation of TRAK in Sparx Systems' Enterprise Architect



Nic Plum

Eclectica Systems Ltd.

www.eclectica-systems.co.uk

This paper is a companion to a presentation made at the INCOSE UK chapter's Annual Systems Engineering Conference made on 8th November 2010.

http://www.incoseonline.org.uk/Program_Files/Calendar/Show_Event_Details.aspx?CatID=Events&EventID=138

It was part of a session titled 'TRAK - An Architecture Framework for Rail'. TRAK itself is not tied to rail or any other domain, however, it is a generic framework which in this case is being used in rail.

CONTENTS

Case Study: An Implementation of TRAK in Sparx Systems' Enterprise Architect

CASE STUDY: AN IMPLEMENTATION OF TRAK IN SPARX SYSTEMS' ENTERPRISE ARCHITECT

Background

During the middle of 2009, London Underground Limited (LUL) decided to use Sparx Systems Enterprise Architect modelling tool (Ref. 1) for the Sub Surface Upgrade Programme (SUP). They had previously been using Microsoft Visio but the overhead in maintaining individual Visio drawings was becoming progressively greater as the set of drawings increased. It was rapidly becoming increasingly impossible as a result to maintain the consistency of the architecture description.

At the same time TRAK itself as a framework (Ref. 2, Ref 3) was being developed whilst architecture descriptions were being delivered into the programme. This was a necessity since there was only a small architecture team. There was therefore a need to make it easier to produce consistent TRAK-compliant views using Sparx Enterprise Architect and to exploit the capabilities of the model repository whilst making the transition from the old Visio approach.

... and this had to be done whilst continuing to deliver architecture descriptions for the SUP since the ITT from LUL still had to be delivered to potential suppliers. This meant that any development had to be slow, incremental and be fitted to the particular demands for architecture view content.

The presentation and this document therefore quickly provide an outline of what was involved and what was learnt whilst doing this in terms of:

- the structure of a Sparx Systems Enterprise Architect extension
- the process of building a release

The Architecture of the Model Driven Generation (MDG) Technology for TRAK

All UML modelling tools have some means of extending the capabilities of the tool. As a minimum they have the ability to import a UML profile which is usually used to provide a set of "template" modelling object and relationship types which are known as stereotypes. Often UML modelling tools will have additional means of extending the behaviour and look and feel of the modelling tool itself. Sparx Systems Enterprise Architect uses an extension method which they refer to as Model Driven Technology (MDG) (Ref. 4). In this way they extend the basic tool to be able to provide diagram types together with object and relationship types so that diagrams for common architecture frameworks can be produced e.g. MODAF, DODAF, TOGAF.

A MDG is itself a UML model in XMI format and stored as XML - in this case it's a model of the TRAK framework. The basic structure is shown in Figure 1.

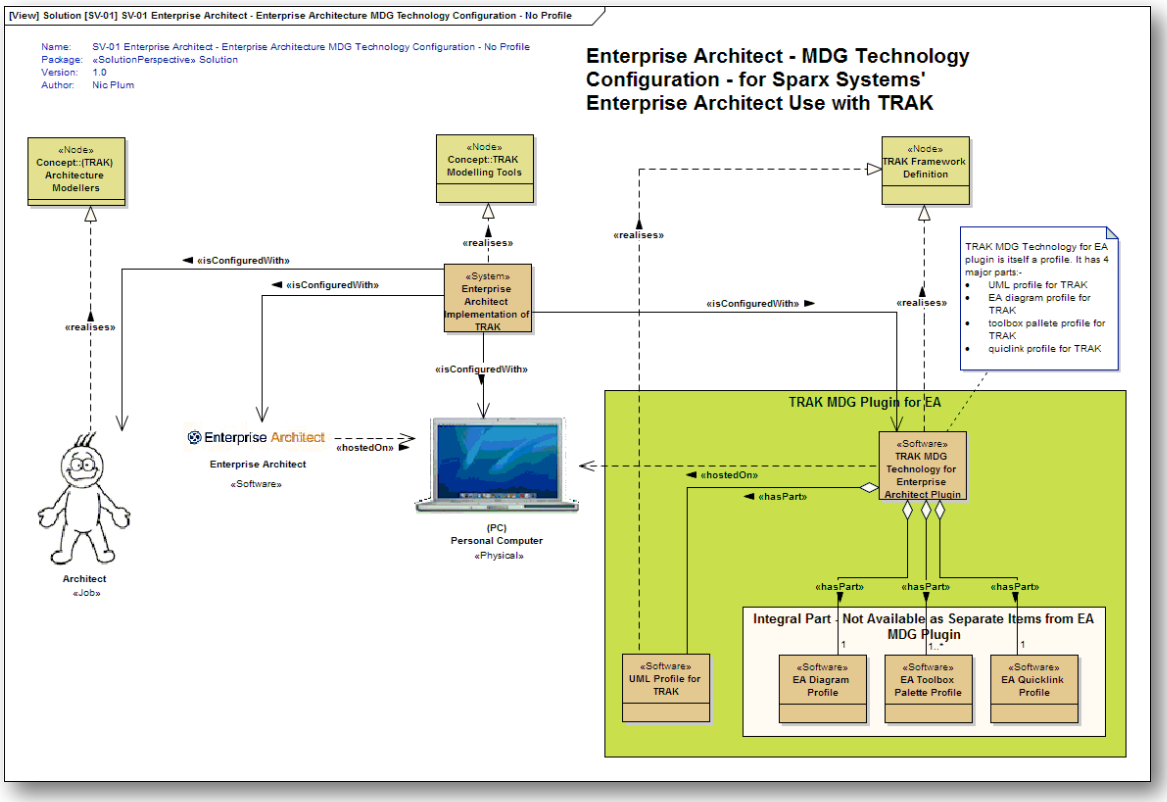


Figure 1 - Structure of the MDG Technology for TRAK

The MDG file is a text file, a non-executable file that sits in a known directory such that when Sparx Systems Enterprise Architect starts it loads the file and makes available the various TRAK modelling objects and views. It also provides ways of modifying the tool behaviour. The MDG includes several other profiles including the UML profile for TRAK that provides the object types used for creating TRAK architecture descriptions. This sits on a computer and the Enterprise Architect software, the MDG plugin for TRAK and the human architect collectively provide the means to implement TRAK in response to the logical (implementation-free) definition of TRAK.

So what's involved in creating the MDG for TRAK?

The Process for Producing the MDG Technology for TRAK

The process for producing the MDG Technology for TRAK includes:

- define the UML implementation of the TRAK metamodel
- define the TRAK viewpoints (diagram) types
- define custom searches
- create TRAK template model
- build and export MDG technology
- release the MDG technology

Define the UML Implementation of the TRAK Metamodel

The architect needs a set of objects which he/she can place on diagrams (views) so we need a representation of the object types and relationships that form the TRAK metamodel (Ref. 3) in a way that Sparx Systems Enterprise Architect can understand. This is achieved by creating a UML profile - another UML model in XMI (XML Metadata Interchange) format (Ref. 5) and saving as a XML file.

For our purposes it is sufficient to have a simple set of objects defined with the specified set of attributes. This is because the metamodel that defines TRAK is not itself in UML - it's a simple diagram with an accompanying table. This is a distinction from MODAF and NAF where THE defining metamodel is itself a UML model (an abstract UML profile) (Ref. 6). This means the UML representation of TRAK needn't be as large or as complex as it is just needs to present or expose the set of object and relationship types.

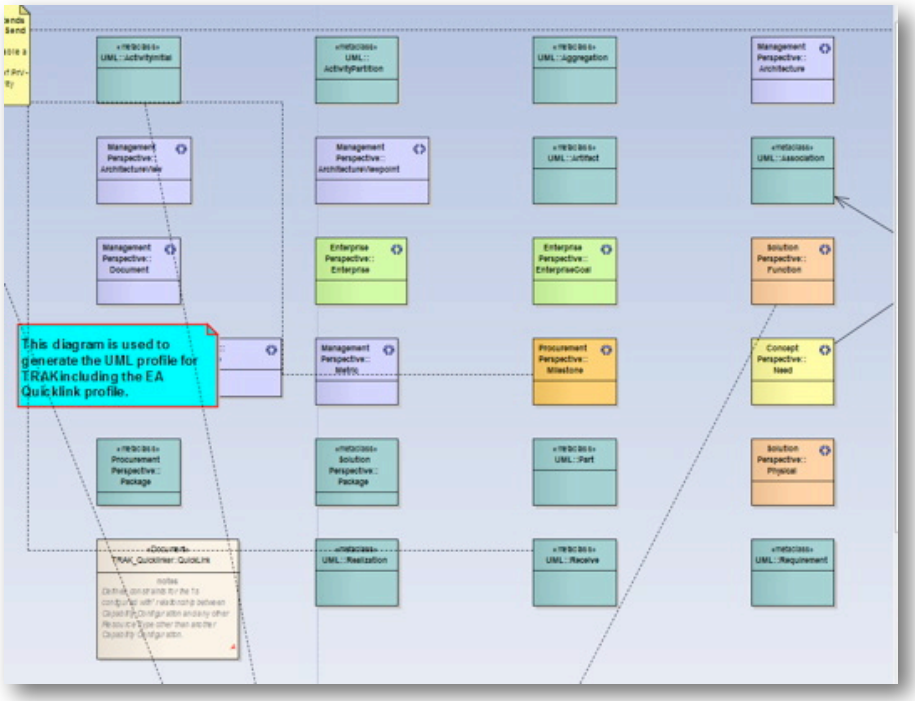


Figure 2 - Export of TRAK UML Profile Achieved from a Single UML Diagram

Not only do we need consistency by providing a common set of modelling objects for the architect but somehow we have to make sure he/she uses the right relationships between the objects and only the allowed objects for the type of TRAK diagram being worked on. Whilst it is possible in Sparx Systems Enterprise Architect to validate the finished diagram it is obviously better to "encourage" the architect to use the right objects in the first place. This has been achieved through 2 approaches - provide the right objects and relationships using custom toolbox palettes and extend the context-sensitive "Quick Linker" (Ref. 7) in Sparx Systems Enterprise Architect that suggests relationships based on the 2 object types being connected and the view type. This uses a definition of allowed and prohibited TRAK relationships that are defined in terms of:

- starting object UML type
- starting object TRAK type
- finish object UML type
- finish object TRAK type
- TRAK view type(s)
- relationship UML type
- TRAK relationship name
- relationship direction
- relationship caption

The allowed and prohibited combinations are obviously quite large. To give some idea of scale the spreadsheet from which this was generated is roughly 700 lines i.e. 700 rules. TRAK has a very small metamodel by comparison with others. The spreadsheet is used to create a CSV text file which is embedded within a UML document artefact and placed on the diagram used to create the TRAK profile.

Having defined the UML object types and Quick Linker relationship constraints that represent the TRAK metamodel these have to be exported using XMI as a XML file. This UML profile for TRAK can then be referred to by other parts of the MDG, for example when listing the appropriate TRAK metamodel object types for use when creating a particular TRAK view (Ref. 2).

Define the TRAK Viewpoints as Diagram Types

The MDG technology also allows us to define custom diagram types as forms of (specialisations of) UML diagram type. For each TRAK view type (defined by the TRAK viewpoints in accordance with ISO 42010) (Ref. 8) that needs to be able to be created there needs to be a definition in terms of its name, description, custom attributes and the type of UML diagram it is based on. Once again, this is achieved through a UML model and the result saved as an individual model as an XML file for each diagram (in EA/UML-speak).

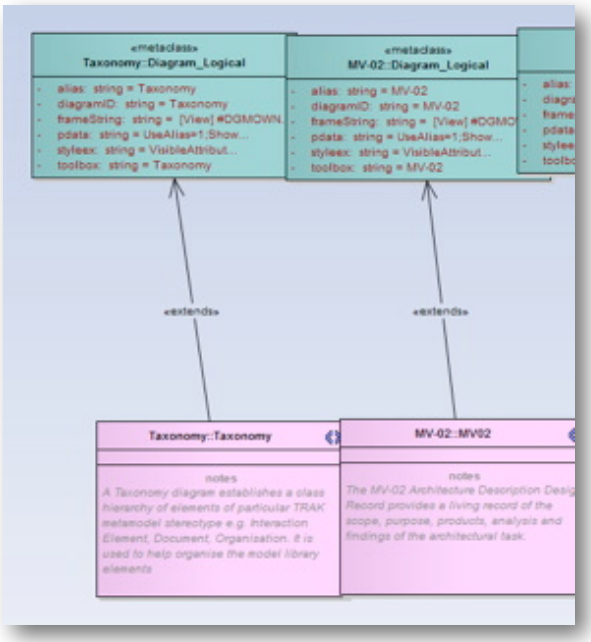


Figure 3 - Definition of a TRAK View Type

Define View-Specific Toolbox Palettes

The architect needs the right set of TRAK object and relationship types from which to pick in order to construct an architecture view. Although it is possible to simply import a UML profile e.g. the one for TRAK which will then display the object and relationship types in a list it is obviously better to present the appropriate set for the particular architecture view type. This is achieved by defining a toolbox palette for each view type and linking the view type to the particular toolbox palette definition.

There are a number of design decisions/considerations made to try and make it easier to choose the right object or relationship type:

- toolbox palettes separated to indicate use. Palettes have been separated to try and separate object types that can be added to any view from those that are view- specific. The object and relationship types which have to be used to construct a view have been placed in a 'Mandatory' toolbox whilst those that are optional but which show additional context have been placed in a 'Optional' toolbox.
- encourage object creation (declaration) on the right TRAK architecture view. In TRAK each metamodel object type has a master architecture view type in which the it is first shown or declared (Ref. 9). For example, resources (systems, software, physical things, organisations, roles and jobs) have first to be added to a SV-01 Solution Structure view before they are used in any other view. Sparx Systems Enterprise Architect won't prevent an object being added. In order, therefore, to show that an object type needs to be or can used but is created and added to another view an icon is displayed in the toolbox palette - . This isn't perfect as it doesn't stop a determined architect from doing the wrong thing, but at least it is a visible indicator of what should be done.

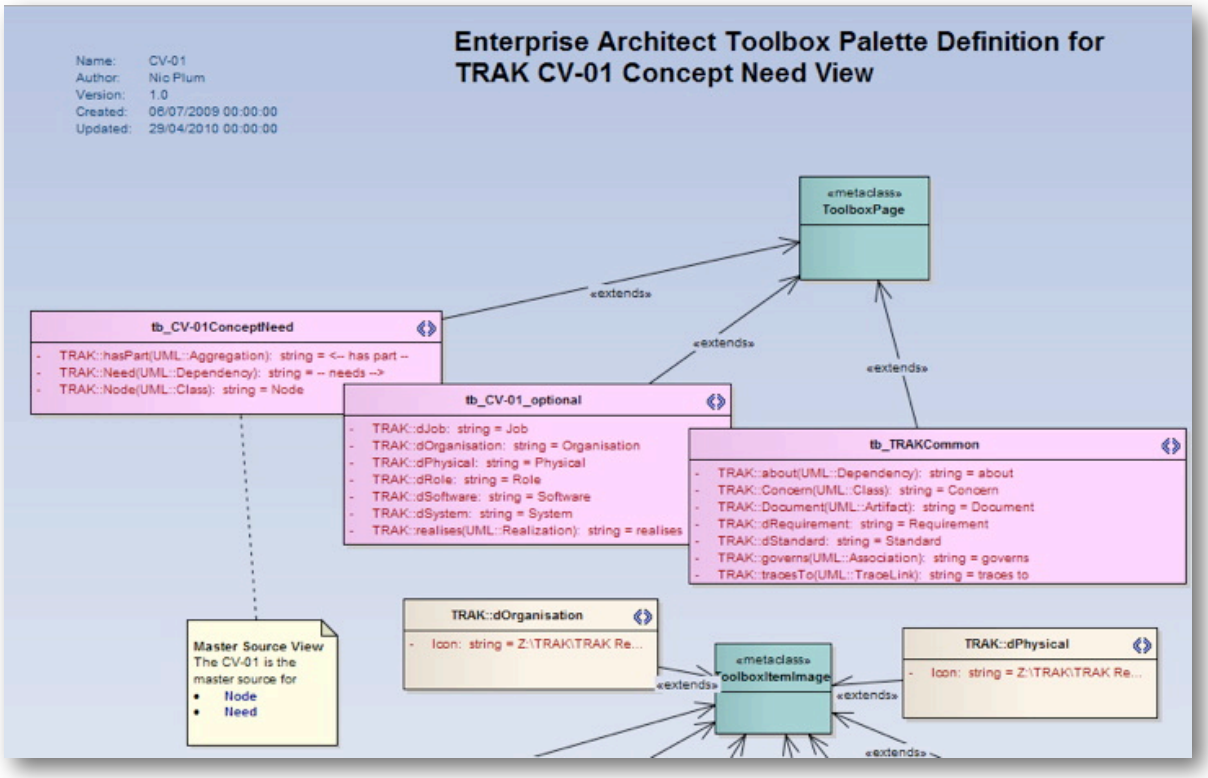


Figure 4 - Definition of a Toolbox Palette

Define Custom Searches

The ability to define custom searches is very useful as it provides the ability to:

- define matrix-like architecture views (usually more compact than connected blocks)
- use relationships established between the objects in an architecture description - to discover, to traverse and to analyse
- manage repository quality

The use of the relationships established is in many ways the raison d'etre of enterprise architecture modelling - the views provide the visualisation of what lies beneath but only one or a few of many ways in which the information can be used and presented. This is where the repository containing many architecture descriptions becomes increasingly useful over time. It also means that methods are needed to spot and correct quality problems to allow it to continue to be useful and correct for others to use i.e. to administer the repository.

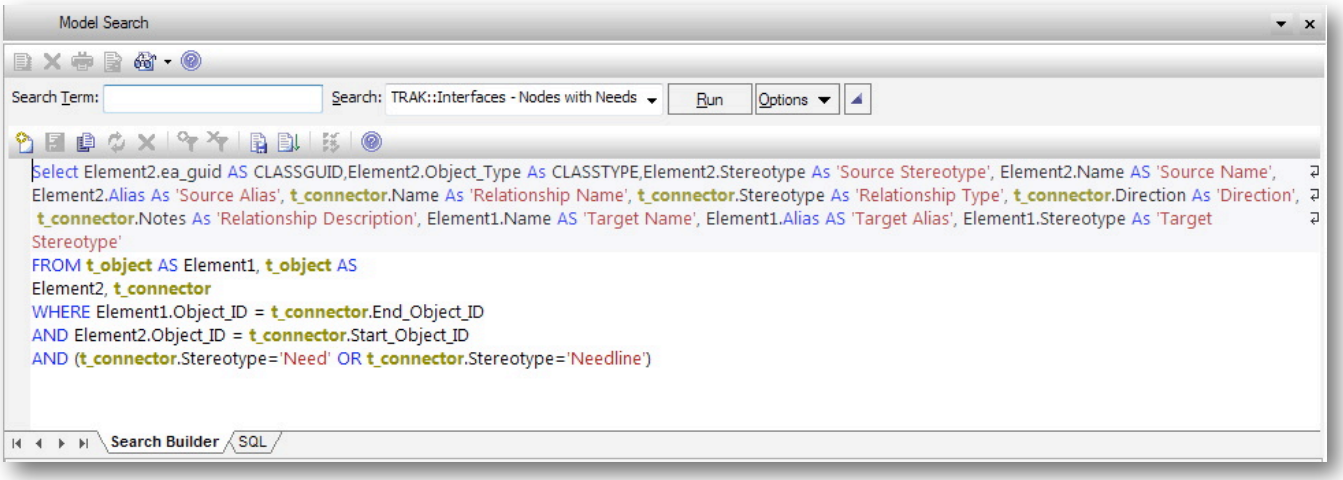


Figure 5 - Definition of a Custom Search Using SQL

Once again the process is that each search is defined and then the set of search definitions is exported as an XML file.

Create Template Model

A sample set of TRAK views is included with the MDG technology. This also provides non-TRAK views to illustrate diagrams that can be used for model management, for example diagrams that provide ways of representing coverage or completeness.

Build and Export the MDG Technology

The last step in creating the MDG technology is to build the MDG technology file. This is achieved using a set of dialogs in a wizard within EA in conjunction with a configuration file (Ref. 10). The result of this is that EA itself welds the various XML files representing the metamodel, quick linker relationships, toolbox palette, diagram and search profiles into a single profile. It also adds basic identification and configuration information.

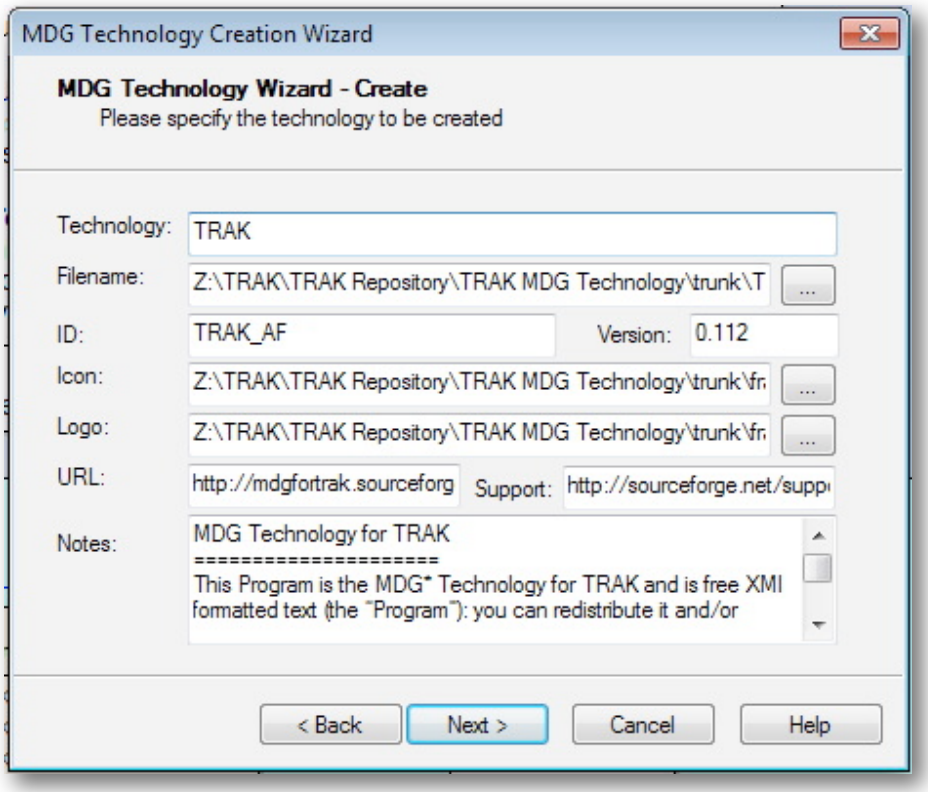


Figure 6 - Wizard Follows Steps Needed to Build MDG From All the Individual Profiles

Release the MDG Technology

The MDG Technology for TRAK is formally released as open source through a project (mdgfortrak) on the Sourceforge website (Ref. 11). Version management is controlled using Subversion (Ref. 12) and feature requests, bugs dealt with using the various tracking databases provided. Release is achieved by updating the central code repository from the local one.

Features

The MDG Technology for TRAK adds the following to the basic Sparx Systems Enterprise Architect modeling tool:

- TRAK architecture views
- view-specific toolbox palettes
- context-sensitive links
- model views
- custom searches

TRAK Architecture Views

The MDG for TRAK adds specific TRAK views to the default New Diagram dialog box.

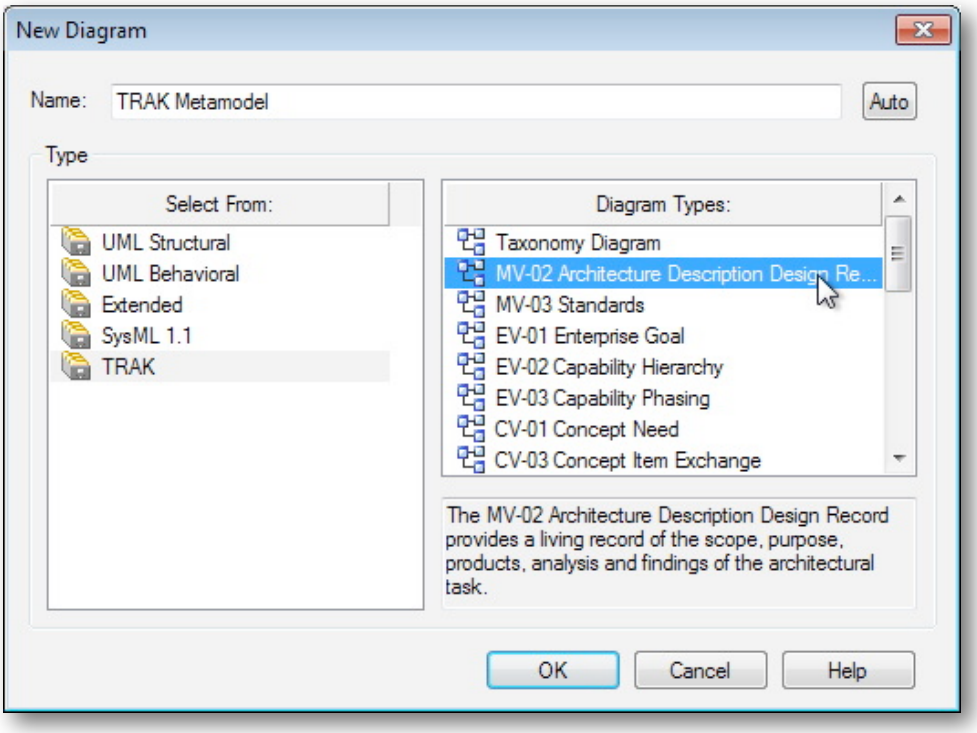


Figure 7 - New Diagram Dialog Showing TRAK Views

View-Specific Toolbox Palettes

Each TRAK view has a set of toolbox palettes

- TRAK Common - TRAK object types that can be added to any view e.g. a requirement, document, concern
- nV-xx - view-specific set of of object types and corresponding relationships. Mandatory ones are those which are needed for the particular view. Optional ones provide added context.

Where there is an icon representing a directory structure this indicates that the object is first created (declared) on a different type of view but used in the current view. In this case, for the SV-01 Solution Structure View, you would show a relationship to the Node being realised by the solution element. Node is first created on the CV-01 Concept Need View. Sparx Systems Enterprise Architect won't actually stop anyone from creating any type of object on any view (without external add-ins) but making it harder to do the wrong thing and easier to do the right thing must improve the consistency and correctness of the resulting architecture description.

Context-Sensitive Links

Sparx Systems Enterprise Architect provides a 'quick linker' method whereby grabbing and dragging an arrow next to a diagram object will cause a pop-up dialog prompting the user to select the type of link to be made. This is extended by the MDG technology for TRAK so that the correct relationships for the types of object being linked are offered. Once selected the relationship is made.

It makes the links in the correct direction irrespective of which object is first selected as the TRAK relationship is made in the direction that the sentence is read e.g. 'has part', 'realises', 'is member of'. Only where the source and target object are the same type is this not possible.

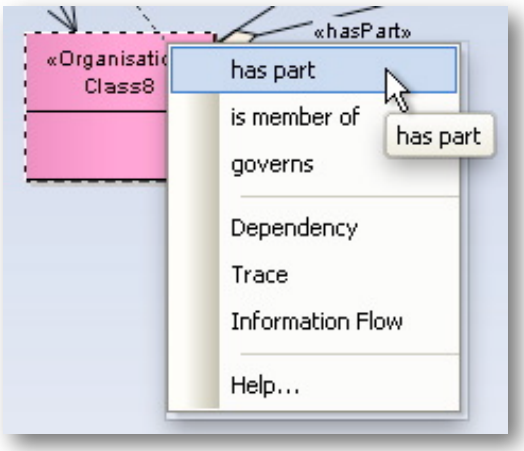


Figure 8 - Context-Sensitive Quick Linker Suggests Correct TRAK Relationships for Objects Being Linked

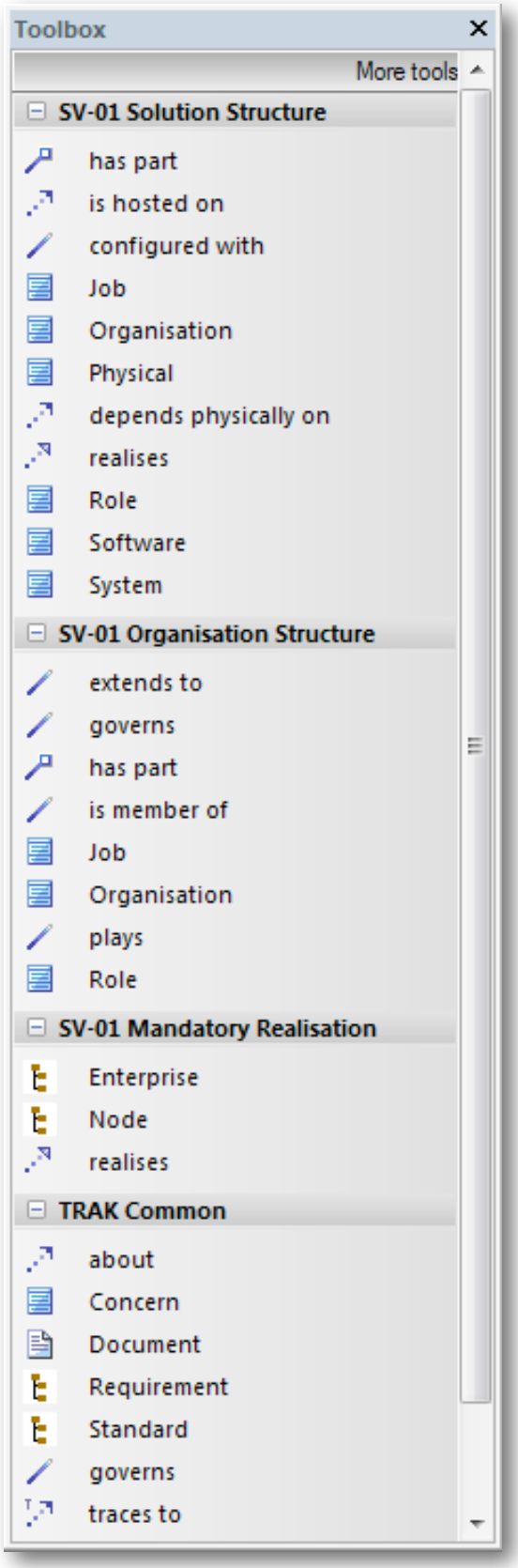


Figure 9 - SV-01 Toolbox Palette Provides All the Object and Relationship Types Needed to Create the SV-01 Solution Structure View

TRAK Model Views

Sparx Systems Enterprise Architect provides model views which are collections of objects from the Sparx project (model repository) listed in a browser. These are based on custom searches and can be used, for example, to notify when the search criteria is met e.g. adding a new diagram.

The TRAK MDG technology uses the custom searches (see definition and also the searches provided) to provide browsable lists of diagrams and object types for navigation and for quality control purposes. This is nothing to do with TRAK as a framework but an aid to repository and model management.

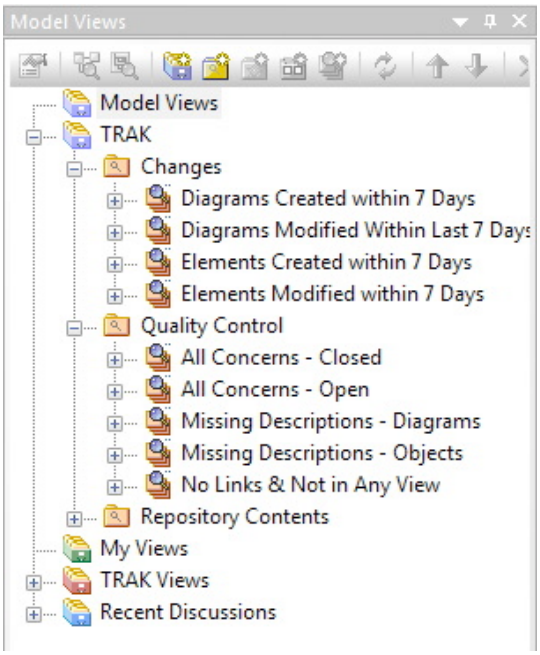


Figure 10 - EA Model Views Allow Repository Diagrams/ Objects Meeting Criteria to be Listed

Custom Searches

Sparx Systems Enterprise Architect provides the ability to run ad hoc and canned searches using the SQL database that stores the architecture description.

The MDG Technology for TRAK extends the basic set of searches and adds the following:

- Solution - Structural Relationships
- Solution - Interactions
- Search Name and Alias
- No Links & Not In Any View
- Missing Description - Objects
- Missing Description - Diagrams
- Interfaces - Nodes with Needs
- Element - Modified in Last 7 Days
- Element - Created in Last 7 Days
- Diagram - Modified in Last 7 Days
- Diagram - Created in Last 7 Days
- Describe All Objects
- Concerns About...
- Concerns - Open
- Concerns - Closed
- All Visions
- All Systems
- All Standards
- All Software
- All Roles
- All Requirements
- All Projects
- All Project Activities
- All Physical
- All Organisations
- All Nodes
- All Milestones
- All Metrics
- All Jobs
- All Items
- All Interaction Elements
- All Functions
- All Enterprises
- All Enterprise Goals
- All Documents
- All Diagrams
- All Contracts
- All Concept Activities
- All Concerns
- All Competenceis
- All Capabilities
- All Architecture Tasks
- All Architecture Descriptions

It also adds some capabilities to search just within selected diagrams (architecture views) in the Sparx Systems Enterprise Architect project browser.

Plans for the Future

The development of the MDG Technology for TRAK as an open source project will lead anywhere there is the user demand or the development interest to do so. The likely areas based on development and usage experience are:

- more complicated queries. Some views, for example the SV-03 Function to Resource Interaction matrix view cannot be directly made in a diagram as it uses graphic object to graphic path mapping. This could be constructed using a SQL query. The potential disadvantage to more complex queries are that they depend on the capabilities of the database engine being used - the project file (.eap) uses a Microsoft Jet engine which has fewer capabilites than, say, a MySQL database engine. If there are different capabilities or SQL syntax it is possible to place the code within a database engine selector () but this makes the query more complex and you need to have access to all of database engines to test the query.
- validation of diagrams. The emphasis has been on trying to prevent errors being created rather than checking on completion. Sparx Systems Enterprise Architect offers the possibility to develop a custom set of TRAK validation rules to check views and, possibly, the entire architecture description. The downside looks to be the extra complexity needed to develop a separate ActiveX COM add-in.
- creation of reading direction indicators for relationships. Relationships created have a link direction but not a reading direction indicator for the relationship. This can be added manually but it would be better for this to be able to be created automatically.

What We Learnt

Quite a lot has been learnt en route.

- incremental development is best - it is suited to situations based on user need e.g. 'I need to be able to construct this type of view'
- constraints are necessary but hard to enforce. Consistency needs to be enforced at all levels - within a view, across an architecture description and applies to the meaning (semantics), presentation and choice. Any tool can at best enforce rules in terms of the right type of element being used correctly on the right architecture views. In practice it's difficult to absolutely prevent people doing silly things and you therefore have to provide additional ways of picking up the worst excesses, hence the quality control searches provided.
- a UML tool is a UML tool and therefore brings with it some expectations. It only understands the world as expressed in terms of UML objects and relationships. For non UML people this can add difficulty, for example in the meaning of the arrow symbols (TRAK mandates that you have to have a text label not just rely on a symbol). This also means that you have to be careful not to embed or impose limitations of UML unnecessarily, for example by choosing the 'wrong' type of diagram as a base e.g. a composite structure diagram will only let you see structure/behaviour within an object. The other danger is that in using a UML tool is that people automatically assume TRAK is like or is a form of UML - it isn't. In the reverse direction they therefore assume that all the UML diagrams will be present in TRAK. It is hard to keep the separation in people's mind between the framework and the representation (implementation) of the framework in a particular language.
- Sparx Systems Enterprise Architect, like any tool, has limitations/quirks.
  - XMI profile creation isn't always modelled consistently e.g. to create a toolbox palette the parent class has specialisations added to it to create attributes for toolbox elements - this is not what you'd expect in a class structure (the specialisations would be added to the child).
  - Sparx Systems Enterprise Architect can model multiple inheritance but can't export multiple inheritance in UML profile. The net result is that the class hierarchy has to be distorted (or ignored) to get the desired set of attributes for any TRAK stereotype
  - the results of searches can't directly be saved as a CSV to make it easy to create tables in a document. There are workarounds using cut and paste and running a script but given that matrix views in Sparx Systems Enterprise Architect do provide this capability it seems inconsistent.
- good support from the tool vendor is essential. There is a relatively small population who do this sort of development and consequently the number of examples to look at is extremely limited. This resulted in quite a lot of 1:1 interaction with Sparx Systems Enterprise Architect support who luckily are pretty good.
- if you're not a dyed-in-the-wool "softie" some things seem very unfamiliar at first. There was a lot of behind the scenes detail within Sparx Systems Enterprise Architect to learn and also getting to grips with the terminology and facilities provided for software development on Sourceforge and use of Subversion.

- getting people comfortable releasing as open source takes a long, long time - there are a lot of misconceptions and getting everyone to the same level of understanding/comfort is hard work.

Conclusions

Extending Sparx Systems Enterprise Architect is pretty straightforward. It could, however, be made easier in some ways by eliminating some of the inconsistencies in approach when defining profiles. It would also be better if it were possible to prevent things when creating diagrams (views) for a profile in the same way that it disallows things when creating UML diagrams rather than relying on validation after the fact. When working to deadlines and having to move quickly onto the next task, it is all too easy to forget or not have time for correction since it's often less important for the immediate customer (single shot task) than for the longer term health and re-use of the repository content.

Providing a standardised set of objects and relationships and trying to constrain architects does produce more correct and consistent models within the limitations of any tool but it doesn't stop bad practices or woolly thinking/organisation so training is a necessary adjunct. Using a proper modelling tool with re-usable objects in a repository is, however, a million times better than a drawing tool like Visio where the maintenance overhead much more quickly becomes unaffordable in trying to keep an architecture description consistent.

Working with a repository and a SQL-based database engine is a must for any size architecture enterprise. Systems engineering and system thinking is primarily about identifying and managing relationships and it therefore relationship-centric. Most tools are software development tools which are primarily object-oriented. The trick with enterprise architecture is exploiting these relationships and therefore the tool must offer ways to exploit and export this information without always relying on visual presentation. This is where future work on the MDG Technology for TRAK will probably continue and this is then as much about repository management as TRAK itself.

References

1. Sparx Systems' Enterprise Architect. http://sparxsystems.com/products/ea/index.html
2. TRAK Enterprise Architecture Viewpoints. http://trakviewpoints.sourceforge.net
3. TRAK Enterprise Architecture Metamodel. http://trakmetamodel.sourceforge.net
4. Sparx Enterprise Architect - MDG Technologies. http://sparxsystems.com/resources/mdg_tech/
5. Wikipedia. XML Metadata Interchange. http://en.wikipedia.org/wiki/XMI
6. MoD. M3 Introduction. http://www.mod.uk/NR/rdonlyres/0360559B-3D6B-4198-BD92-D2B783A7E831/0/20090310_MODAF_Meta_Model_V1_0_U.pdf
7. Sparx Enterprise Architect. The Quick Linker. http://www.sparxsystems.com/enterprise_architect_user_guide/modeling_fundamentals/quick_links.html
8. ISO/IEC 42010:2007. Systems and software engineering -- Recommended practice for architectural description of software-intensive systems. http://www.iso-architecture.org/ieee-1471/
9. TRAK Community Site. TRAK::Master Architecture View for Each Stereotype. http://trak-community.org/index.php/wiki/TRAK:Master_Architecture_View_for_Each_Stereotype
10. Sparx Systems. Create MDG Technology http://www.sparxsystems.com/enterprise_architect_user_guide/modeling_languages/creatingmdgtechnologies.html
11. Sourceforge. MDG for TRAK Project. http://mdgfortrak.sourceforge.net
12. Apache. Subversion Features. http://subversion.apache.org/features.html