

INTSPEI P-Explorer: integration tool for software projects

Vladimir Pavlov¹, Anatoliy Doroshenko¹, Petro Protsyk¹, Tatyana Taganskaya¹,
Victor Sergienko¹

¹ INTSPEI, 2nd floor, 4 Mykoly Hrinchenko Street, 03038 Kiev, Ukraine
{vpavlov, doroshenko, protsyk, taganskaya, vsergienko}@intspei.com

Abstract. This paper presents a new tool for software project management – INTSPEI P-Explorer. It is based on the original paradigm of P-Modeling Framework developed by the International Software Productivity and Engineering Institute (INTSPEI) that is intended to specifically provide a new, higher level in software development. The major benefit of INTSPEI P-Explorer is bringing a holistic view of a project.

Keywords: integration tool for software projects, project elements and their attributes, project elements relations, computation on attributes.

1 Introduction

The goal of this article is to present a new tool for software project managers – INTSPEI P-Explorer. It is based on the original paradigm of INTSPEI P-Modeling Framework [1] and will help to stay in touch with the integral status of the project; manage dependencies between elements and work with different kinds of deliverables using the same environment.

The purpose of INTSPEI P-Explorer is to process all types of project information in the single application. This information is hierarchically structured according to its location. However it could be represented in a different manner using Views and Filters. View – is a way to show data to the user. Filters are the means for selecting the data according to criteria specified to show them in views.

The major benefit of INTSPEI P-Explorer is bringing a holistic view of the overall project. Project information comes from different sources and is tracked by different systems. For example, requirements are sent by the customer in a MS Word document and are stored in the Requirements Management System; source code is stored in the Source Control Repository; and System Design is written in UML diagrams, etc. Project manager needs different systems to view and manage this heterogeneous information and this often affects productivity. INTSPEI P-Explorer resolves all these issues without integration into any existing software process.

INTSPEI P-Explorer is not only able to store project information, but also to work within its structure: That is, every item could be decomposed into immediate constituents such as attributes and sub-items. For example a UML model may be decomposed into the list of diagrams, diagrams may contain actors, classes, packages, associations, etc. Moreover, such functionality may be extended by the use of special plug-ins which will provide the means of decomposing new items.

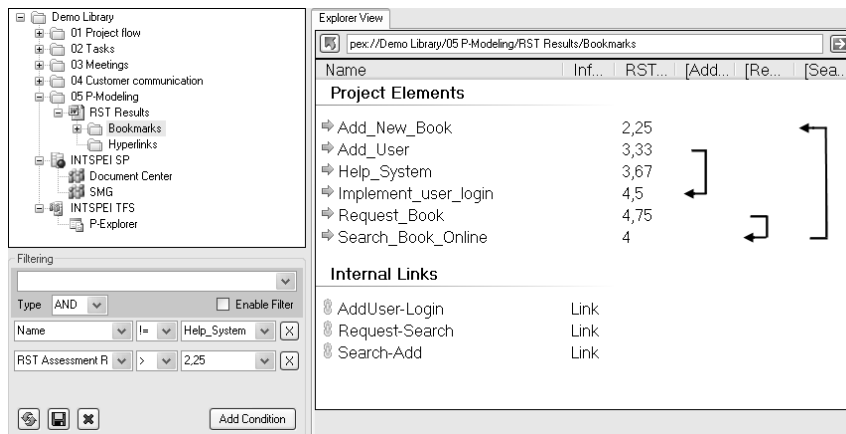


Fig. 1. INTSPEI P-Explorer Main Window.

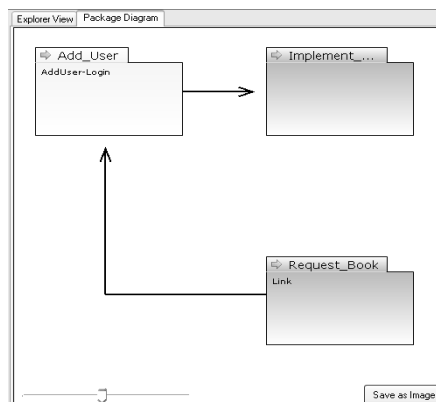


Fig. 2. View relations.

There is one very special feature of INTSPEI P-Explorer: as The Traceability Management Tool. While traceability is recognized as a “critical success factor” in software development [2], there is a lack of effective software solutions [3, 4]. Problems are caused by a variety of different artifacts, levels of their formalization, and systems involved in the development process. Here, one of the most distinguishing features of INTSPEI P-Explorer is that it can establish traceability links among artifacts created in different systems. Usually tools allow creating links only between artifacts stored inside the tool itself. Additionally, different traceability

metrics may be computed for the top level links according to the parent to children links.

2 Concepts

There are several concepts which lay in the theoretical foundations of INTSPEI P-Explorer. They are: “project tree”, “project element”, “attribute” and “relation”.

The simplest one is an attribute. This is a pair with a name and value. The attributes are used to describe project element. By the project element we mean anything related to the project: documents, requirements, diagrams, source code, and all relations between them, etc. The notion of project element is very close to that of artifact. The difference is shown that the artifact, as a rule, is understood as a physically existing object such as a document, or a file on the disk, etc. A project element also includes “virtual” entities such as abstract relations. Inside the tool, the project element is described by a set of attributes. The values of some attributes may be computed using values of other attributes. Such attributes are called computable attributes. Project elements are organized into a tree structure to create a hierarchy. This structure is called a Project Tree. A project element also has a structure by itself, e.g. it may include other project elements as sub-items.

Now, we would like to consider an example of computable attribute. Suppose there is a tree of tasks with sub-tasks:

- Tasks (completeness = Average value for all tasks)
 - Task1 (completeness = 50%)
 - Task2 (completeness = ?)
 - Task 2.1 (completeness = 5%)
 - Task 2.2 (completeness = 15%)

We need to evaluate the average state of completeness attribute for all tasks and their sub-tasks. First of all, the value of the completeness attribute must be evaluated for the “Task 2”, which is 10%. Then the average completeness for this example will be “30%”. In INTSPEI P-Explorer the formula for such calculations is simple: $AVG(chilids, \text{“completeness”})$, where “chilids” – denotes all sub elements of the current element and “completeness” is a name of the attribute. In the case when the value of “completeness” attribute is undefined, this formula will be applied to the sub-items of the corresponding element to calculate it.

INTSPEI P-Explorer possesses a rich language which allows writing custom formulas to calculate different metrics. There are also predefined set of common functions: average, sum, max, etc. The language is able to operate with whole project tree, individual project elements, relations between them and attributes.

There is a special type of project element called “Relation.” It is used to denote the most abstract unidirectional relationship between two project elements. Here, abstract means that by default the relationship doesn’t have any special meaning. This meaning is given by attributes. There is a predefined attribute called “Type”, which is used to define the type of the relationship. For example: Dependency, Traceability,

Composition, etc. There are may be other attributes for example: “value” - which denotes the quality of semantic traceability between two elements [1].

3 Use cases

Here we would like to describe a few possible use cases of INTSPEI P-Explorer in managing software projects:

- Establish traceability relations between project elements like: requirements, design elements, source code and test cases;
- Store and calculate project metrics: requirements coverage with test cases and vice versa, source coverage with requirements, completeness, etc;
- Track project schedule and quality;
- Access all project elements stored in different systems from a single integration point.

4 Conclusions

INTSPEI P-Explorer features a number of original ideas together with underlying methodological basis. It brings a new way to manage projects through:

- Single storage for project elements of different nature,
- Ability to create, import and manage attributes for each project element,
- Ability to store relations between any two project elements,
- Performing calculations on the project tree,
- Different views and filters for representing graphical information,
- Straightforward integration with any third party system with plug-ins,

The role of INTSPEI P-Explorer may be understood as a control center for software projects. We believe that application of INTSPEI P-Explorer in software development may significantly increase productivity of the managers.

References

1. INTSPEI P-Modeling Framework Whitepaper, INTSPEI, <http://www.intspei.com>.
2. Domges, R. and Pohl, K. Adapting Traceability Environments to Project Specific Needs. *CACM*. 41(12), p. 54-62, 1998.
3. Spanoudakis, G. and Zisman, A. *Software Traceability: A Roadmap* Advances in Software Engineering and Knowledge Engineering. Chang, S.K. ed. 3, World Scientific Publishing, 2005.
4. O. C. Z. Gotel and A. C. W. Finkelstein, “An Analysis of the Requirements Traceability Problem,” Proceedings of the First International Conference on Requirements Engineering, Utrecht, The Netherlands (1994), pp. 94–101.
5. INTSPEI P-Explorer, INTSPEI, <http://www.intspei.com/products/pexplorer.aspx>

INTSPEI P-Explorer: integration tool for software projects

(Demo part)

Vladimir Pavlov¹, Anatoliy Doroshenko¹, Petro Protsyk¹, Tatyana Taganskaya¹,
Victor Sergienko¹

¹ INTSPEI, 2nd floor, 4 Mykoly Hrinchenko Street, 03038 Kiev, Ukraine
{vpavlov, doroshenko, protsyk, taganskaya, vsergienko@intspei.com}

The demo part of the paper presents a sample project installed with INTSPEI P-Explorer out-of-box and essential parts of a real customer case study.

1. Demonstration of out-of-the-box sample

Demo sample of INTSPEI P-Explorer includes:

- Sample folder structure for software process;
- Artifacts (files) for:
 - o Requirements (Microsoft Word documents);
 - o Design (UML diagram in XMI [2]);
 - o Code (a C# project)
 - o Tests – manual (Microsoft Word documents);
- Traceability relations in INTSPEI P-Explorer database;
- Traceability attributes added to project folders and artifacts.

2. Demonstration of a real project case study

We depict some parts, essential from manager's point of view, of a real project case study. Our customer, particularly, had the following setup:

- Requirements are kept in DOORS [3];

Test cases (mostly manual, some automated) are kept in internal test software.

This case study appears to depict some typical problems experienced in software project engineering, namely:

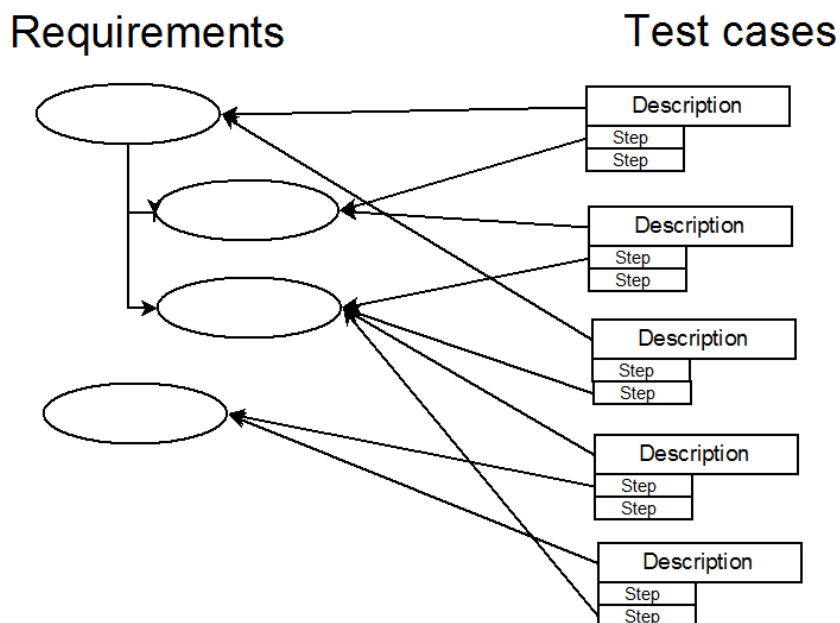
Requirements, development tasks and test cases have been desynchronized;

Requirements leaked to test cases: components of the requirements appeared in the test cases only.

Requirements and test cases sets are incomplete themselves as they complemented each other. Sometimes this is a conscious decision because fixing it required too large of an effort to restructure both requirements and test cases.

Test cases are very long-sized and concern main requirement only; but actually cover many requirements simultaneously (multiple-to-multiple type of relation). So tests can be duplicated and task implementations can be delayed because of reworks.

Problems in requirements and test cases relationship can be illustrated with the following picture:



What we have done with INTSPEI P-Explorer is providing better traceability features under conditions that appeared to be needed in the project. As a result:

1. DOORS backend plug-in is created by INTSPEI as a part of INTSPEI P-Explorer 2.0 release.
2. Internal test tracker backend plug-in is created by third-party contractor for customer.

The back-end plug-in is a third-party component installed into INTSPEI P-Explorer providing an interface to connect to that system. Backend plugin requires very little code, so integration with external system can be completed in very short terms, provided the system is open to integration.

The minimal API to implement for external backend plug-in is shown with following software demonstrations:

Demo 1.

```

public interface IPlugin
{
    bool Populate(ProjectElement itm);

    string[] GetFilterExtension();

    bool AppliesToNode(ProjectElement item);
    bool AppliesToChildren(ProjectElement item);

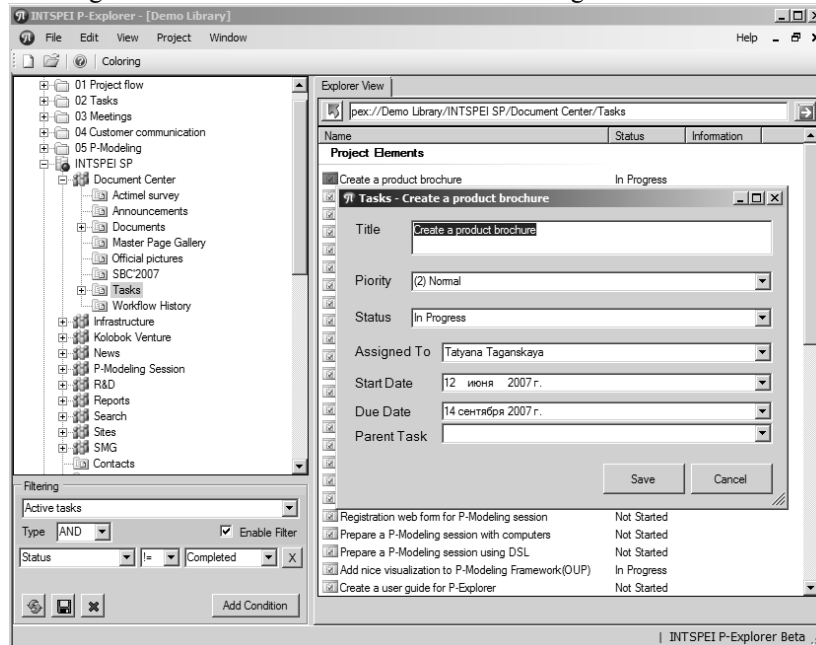
    string GetDisplayName();
}

public interface INodeFactory
{
    ProjectElement createTreeNode(int ID, ProjectTree tree, string ElementType);
    ICommand[] GetInsertMenuItems();
}

```

DEMO 2:

1. Starting INTSPEI P-Explorer on a computer with pre-installed environment.
2. Creating a new project.
3. Connecting to DOORS backend and adding a DOORS-node to project.
4. Browsing DOORS requirements with INTSPEI P-Explorer.
5. Connecting to MS SharePoint backend – adding corresponding project subtree;
6. Browsing MS SharePoint task list. Edit a task – setting task’s status.



In order to establish a traceability relationship between requirements and test cases: Establish it with Drag-n-Drop INTSPEI P-Explorer GUI; needed sequence of operations is shown in:

DEMO 3:

1. Opening SharePoint list in INTSPEI P-Explorer.
2. Selecting several work items.
3. Drag-n-Dropping them onto DOORS requirement.
4. Selecting “Create Relationship” item From a pop-up menu and creating traceability relationships.
5. In the same manner, linking several Microsoft Word document subchapters to the MS SharePoint work item.

This clearly illustrates INTSPEI P-Explorer’s extensibility and openness to being used with different architectures. INTSPEI P-Explorer *does not* require any development lifecycle (SDLC) changes or software changes (with the possible exception of needing some backend plug-ins), and moreover, helps to optimize SDLC for software being developed.

3. Traceability coverage measurement and other metrics

Traceability metrics calculation is built into the INTSPEI P-Explorer first order formulae. These facilities are shown in following demos:

DEMO 4:

1. Opening pre-installed project including DOORS requirements and Microsoft Word test cases.
2. Demonstrating traceability links.
3. Demonstrating traceability formulae in project attributes.
4. Calculating an attribute and demonstrating requirements coverage measurement.

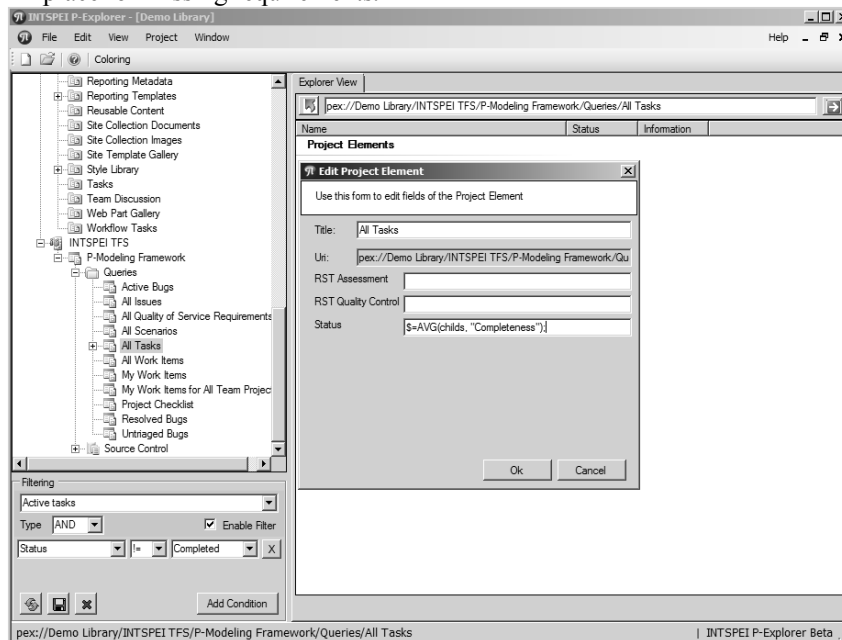
Combined with programmability, traceability tool allows for measurement of different project metrics:

- Measuring percentage of requirements covered by test cases;
- Measuring the volume of work to regroup test cases' steps in order for one test case to cover only one requirement;
- Finding test case steps not covered by any requirement and vice versa;
- Importing a DOORS requirement reference from SoftTest (their internal task&test tracker);
- Finding requirement and test case coverage for program code:
 - Showing what code will change if certain requirement set changes;
 - Showing what test cases need changes if requirements change.

DEMO 5:

- opening Requirements Project Node;

- demonstrating test coverage attribute formula;
- using a filter to see only requirements not covered by tests;
- using a filter to see only over-covered requirements;
- using a filter to see test components not related to any requirements: a potential place for missing requirements.



4. Other tool capabilities demos

Demo 6: UML tools integration:

- opening XMI file saved in UML tool, like StarUml [4];
- creating a relationship to requirement;
- creating a relationship to source code.

Demo 7: integration with MS SharePoint:

- opening a work items list, filter it;
- opening task's properties, marking the task as completed.

Demo 8: integration with TFS.

- browsing requirements;
- creating a relation from requirement to source code;
- updating requirement text.

5. References

1. INTSPEI P-Modeling Framework Whitepaper, INTSPEI, <http://www.intspei.com>.
2. Object management group (OMG) standard for exchanging metadata information via Extensible Markup Language (XML) - interchange format for UML models, <http://www.omg.org/technology/documents/formal/xmi.htm>
3. DOORS. Requirements management system by Telelogic - <http://www.telelogic.com/products/doors/>
4. StarUml: Open-source UML/MDA platform, <http://staruml.com/>