

## ValidationBench

The key to the power of many of iTQP's software quality and validation offerings can be directly traced to our overall project delivery philosophy. This philosophy, based on the experience of literally thousands of projects, is embodied in our proprietary software engineering information model called [Glue](#).

A specific subset of this model, called ValidationBench, has been implemented on a variety of platforms, including as a DOORS™ configuration and its associated add-ins. (DOORS is a product of [Telelogic, AB.](#))

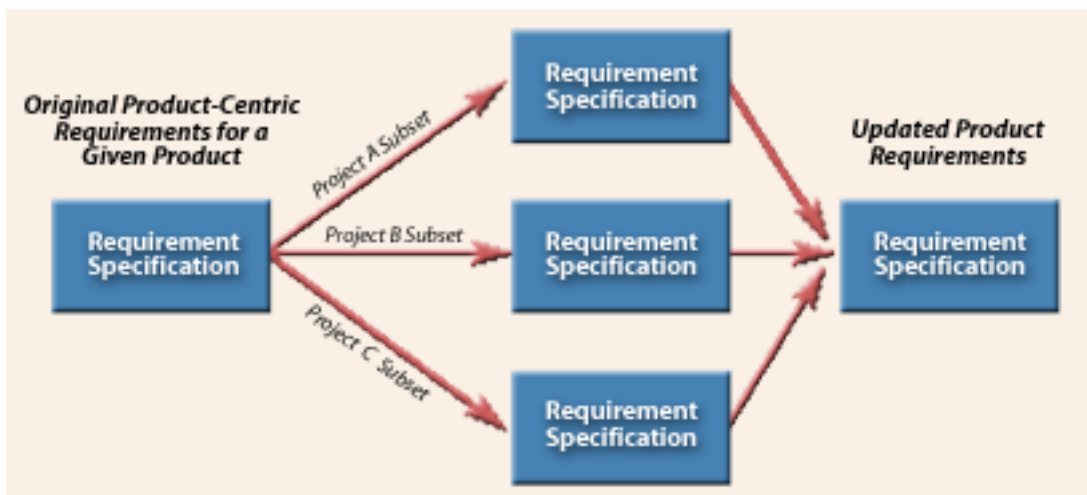
ValidationBench manages the development, maintenance, analysis, and completion status of all four primary technology project artifacts: Problem, solution, delivery, and validation. Moreover, the tool also actively manages all the connections, links, and associations among these artifacts.

Specifically, the DOORS implementation of ValidationBench consists of the following DOORS components:

- A set of DOORS *formal* and *link modules*, together with the associated *link sets* and business rules, that completely implements the Glue software engineering information model
- Object level *attributes* for specifying object instance properties and supporting iTQP's proprietary software engineering and risk management metrics
- *Templates* for consistent production of all artifacts and work products
- Custom *filters* and *views* to facilitate knowledge worker data entry, analysis, and problem solving
- Dynamic *DXL attributes* for real-time computation of completion and validation status for all objects and artifacts
- *DXL user functions* that implement special ValidationBench power-user actions: seamless closed loop work-flow integration of common external testing tools (Segue, CompuWare, Mercury, etc.), tree-view packaging plan status, automatic creation of integration and acceptance iterations for error free builds and reliable testing, etc.
- *DXL user functions* for upload of software engineering metrics to standard dashboard facilities
- *Source code management integration* to coordinate code changes with the development, validation, and delivery of packages

The DOORS tool has been a leader for decades in requirements management, and this emphasis, coupled with iTQP's requirements-based validation focus makes this an exceedingly effective combination for our customers. (And, for us, too, since all of iTQP's systems integration engagements are managed with ValidationBench).

For example, the figure below illustrates the use of the DOORS technology to facilitate the management of *business centric* or *product centric* requirements over multiple projects and across multiple overlapping time horizons.



This is an important issue for managing, leveraging, and delivering business value across the enterprise. With the product centric approach, a set of requirements that define particular business functions (order entry, claims, etc.) or product sets (toaster oven, standard auto insurance, etc.) are captured and refined once. These product centric requirements then are managed collectively as an entity (much like any asset in the company) so that this bundle of integrated requirements is continually being synchronized with its real-life operational counterpart. As the business evolves, the underlying requirements that characterize that business can be altered accordingly, so that an organization always has a clear statement of the needs, relationships, and attributes of its important operating elements.

As can be seen from the diagram, ValidationBench allows each project to select the specific subset of requirements relevant to a given project's scope and then manages the interaction among the various project teams to ensure that an integrated and complete product specification is preserved across time and across the enterprise.

For example, one project might be developing a major extension to a product's capability, while another effort might be simply carrying out routine maintenance for that same product. In all cases, this business architectural approach ensures that each team is presented with a common set of business requirements for the product.

Moreover, since ValidationBench manages the links between a given set of requirements and its underlying software implementation (i.e., via the Supporting Specification link), developers will find it easier to reuse existing solutions, rather than rebuilding. Accordingly, *component re-use* is a central principle of the Glue architecture, and in particular, this direct linkage between a specific software component and its underlying requirements dramatically improves re-use in practice. All of which significantly reduces defects and time to market.

Another important aspect of the Glue architecture (and thus ValidationBench) is its integration with key repositories and tool sets.

Three important classes of these technologies stand out:

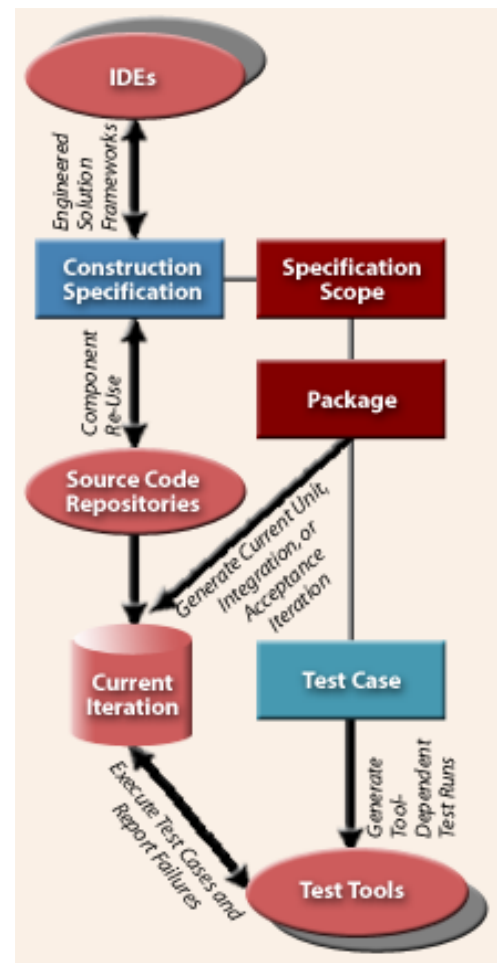
- IDE/Design tools
- Source code repositories
- Testing tools

While DOORS itself can serve as a repository for virtually any type of artifact or work product, our software engineering architecture promotes best of breed solutions that can be integrated into a comprehensive platform for managing technology delivery. The DOORS product facilitates this best of breed approach.

The second example shown below illustrates how ValidationBench dynamically manages the interactions among these key classes of development technologies.

Three aspects of this mechanism are especially noteworthy:

- The linkage of the various development and modeling tools directly with a source code repository to preserve source integrity and to simplify component re-use as mentioned above
- The ability for ValidationBench to automatically generate, at any time, the software manifest for the current development iteration based on dynamically evaluating each package's completion status and its corresponding functional scope; this deterministic and programmatic mechanism dramatically reduces errors and, in addition, guides the validation team by highlighting all the packages that must be validated by indicating which specific test plans must be executed in order to advance the project to its next implementation stage (regression testing is also simplified, since the package boundaries are natural fall-back regression points, which optimizes testing resources)
- The direct interaction between ValidationBench and various testing tools to provide a complete closed-loop automated validation process, from test case design, test run execution, failure reporting, root cause analysis, defect identification, regression testing, and the automatic closing of resolved failures



---

In summary, ValidationBench benefits our customers in the following manner:

- Manages all key project delivery objects, from initial conceptualization to final implementation
- Proprietary value delivery process allows project manager to aggressively manage risk by dynamically configuring packaging definitions and delivery sequences, as needed, throughout the effort
- Completely independent of client project management system, development environment, or tool set
- Coordinates the deployment, distribution, deliverable production, and completion status of independent work packets that allow different workers in different geographic locations to efficiently collaborate on a single project
- Dynamically generates samples and deliverable frameworks as needed to support work groups and facilitate the efficient production of high quality, pre-tested work products
- Integrates a wide variety of deliverable production tools and development environments for direct inter-operability and increased productivity
- Automatically captures key project metrics for later use and analysis—for example, continually provides up to date status on the primary metric that governs the true progress of all technology delivery engagements: *How many requirements have successfully been implemented?*

**For more information, please contact us at (630) 365-1606, or visit [www.itestqp.com](http://www.itestqp.com).**