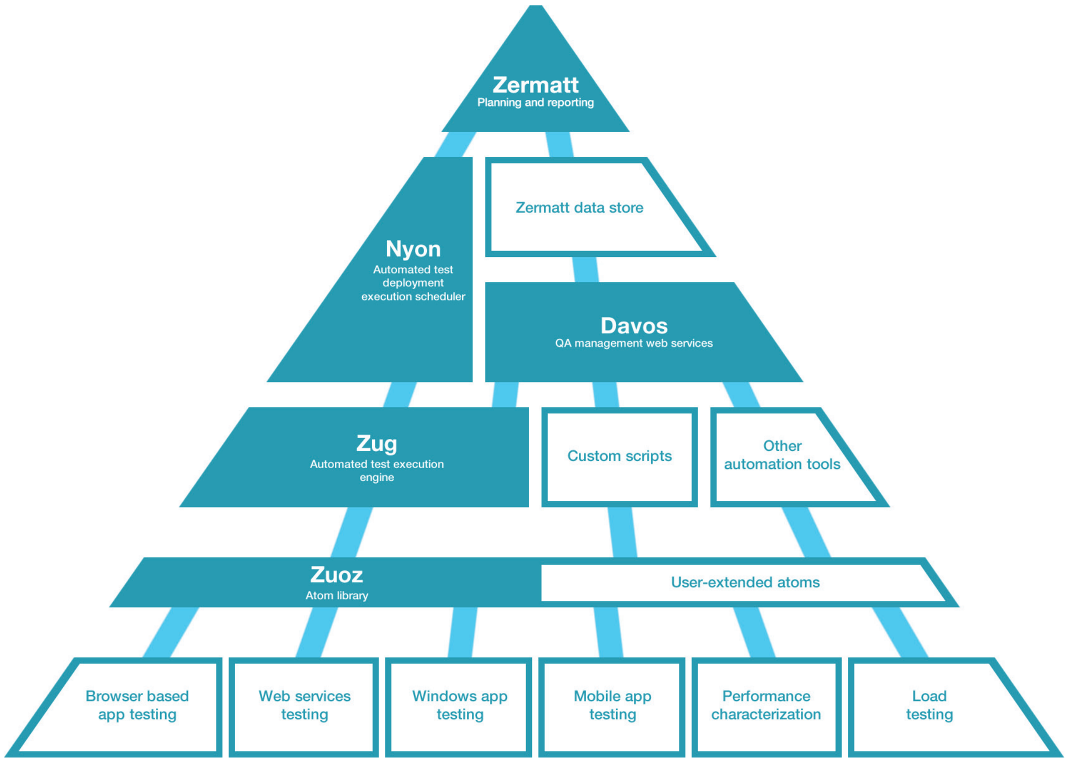
BF-273
Salt Lake City
Kolkata-7020064
India

The diagram shows an architectural view of how the different Automature products fit together to provide a comprehensive set of capabilities needed to run a successful, effective, and lean software quality assurance organization.



At the center of the architecture is Automature's unique data store, which keeps track of all aspects of the information needed to run a QA function. Included is not only the results of test cases, but also other attributes of those tests, that provide a more comprehensive understanding of their purpose, their execution history, their execution environment, their unique combinations of test variables, their relationships to the requirements. Also, provided for in the data model, is an ability to organize test cases and requirements in a multitude of customizable ways, that assist in test planning.

Access to this data store is provided in two ways, viz. through a browser based interactive interface called ZERMATT, and programmatically through a webservices interface called DAVOS.

DAVOS provides the ability for automation engines to interact with the data store using higher level abstractions, and packaging them as services. This allows third party automation tools, and especially custom test harnesses to easily take advantage of the advanced planning and reporting functionality provided by ZERMATT, without the need to write and maintain such a comprehensive and complex application. Automature's own automation engine leverages this interface to store execution data.

ZERMATT is Automature's planning and reporting tool, that acts as a cockpit for managing all software QA functions. It is designed to make the book-keeping tasks of the QA director a breeze, and free up time to strategize about more comprehensive ways to find bugs, thus leading to more yield on investments in quality. ZERMATT is also unique in that it, consciously as a design goal, attempts to provide relevant dashboards targeted to all levels of the company, and not just the engineering organization. This approach promotes quality as a pervasive obsession across the entire staff. ZERMATT, architecturally, is based on the field-tested, widely used TWiki platform, an open-source, function-rich collaboration tool. ZERMATT can run on both Windows, as well as Linux platforms.

ZUG is Automature's Java-based test execution engine. It supports a spreadsheet based test specification language, which is simple enough for non-programmers to comprehend, but is rich enough to support multi-threaded execution of testcases, or individual steps within them. The architecture of the language leverages the concept of atoms and molecules, described in http://www.scribd.com/doc/49785507/97/Test-Atoms-and-Test-Molecules by James Whittaker in his book EXPLORATORY SOFTWARE TESTING. ZUG leverages DAVOS to interact with the data store, keeping information about the execution organized, without having to concern itself with the actual datamodel, which can sometimes be a moving target. ZUG is also architected to work independently of the data store, a capability that is needed while debugging the test specification itself. By virtue of its Java implementation, ZUG can be deployed on any Java supported platform. It has been tested on Linux and Windows.

ZUOZ is Automature's extensible set of atoms, organized by application domain. ZUOZ adds capabilities to application specific testing. ZUOZ atoms can be implemented in any language supported on the target platform. ZUOZ atoms interact with ZUG through a small well-defined API. ZUG also supports an array of built-in atoms, that execute in-process (and are therefore more efficient).

NYON is an addon product for ZUG, allowing individual test suites to be scheduled and executed on one or more target machines. NYON cooperates with ZERMATT and ZUG to initiate test cycles unattended.