XIMEDES

Fit for purpose, use and change
For most organizations today, executing business strategy and developing new business opportunities is intricately linked with the capabilities offered by information technology. Day-to-day operations are highly dependent on existing ICT infrastructure and applications, while developing new business unavoidably requires investing in ICT capabilities. Ximedes helps organizations achieve their business goals by creating bespoke software products, fit for use and purpose, whose elegant design and quality of implementation make them capable of supporting multi-year business cases in an ever-changing world.

In this whitepaper we describe the Ximedes approach for software development, delivering high-quality software, maintenance and operational support, during the complete lifecycle of your product.
**What is ‘good’ software?**

Evaluating if an existing software product or cloud service will be capable of effectively supporting your business is not an easy task. Do the listed features match with your requirements? (And how can you tell simply from a list of features?) Will your users be able to work with it, without being overwhelmed with all kinds of screens and functions that have no clear relationship with the task they need to perform? Are your system administrators capable of configuring and hosting the software reliably in your existing ICT infrastructure?

These questions can be categorized under three, equally important, fitness criteria: fitness for purpose, fitness for use, and fitness for change. In the next sections, we will explain how the Ximedes approach to software development addresses these questions.

**Fit for purpose**

Software is considered fit for purpose if it offers sufficient functionality to achieve its intended business purpose. Software can be unfit for purpose because of some defect in its implementation, or because it simply does not offer the functionality needed.

Developing software that is fit for purpose starts with obtaining a thorough understanding of that purpose in terms of the business processes it needs to support. Note that this purpose is typically not technical, but relates to some business goal. Translating this high-level purpose into SMART\(^1\) functional and non-functional software requirements is the most important purpose of a definition phase – the first phase of any Ximedes development project.

These requirements are the basis for software design and implementation, as well as the many types of software test cases. Validating whether software is fit for purpose essentially means validating that the functional requirements have been met. Ximedes uses the TMAP approach to software testing to determine a clear test strategy and execution framework.

We take great effort to automate the execution of test cases as much as possible, running the whole set of tests several times each day to make sure that functional issues (i.e. bugs) are detected as early as possible, even before the software has been deployed to any test environment. Automated tests are of crucial importance to deliver new versions of the software as swiftly as possible, without compromising the quality needed to support your business.

---

\(^1\) **Specific, Measurable, Assignable, Realistic, Time-related**

– see http://en.wikipedia.org/wiki/SMART_criteria

---

**Image 1 - Every app must be Fit for Purpose, Use and Change**

“A simple architecture minimises the cost of change”
Fitness for purpose, however, is not reached with success—fully passing all test cases. In practice, it is very common that users change their minds about the purpose and functional requirements of their software once a first version is delivered. For this reason, Ximedes advises most clients to be participants in an agile development effort (such as SCRUM), which focuses on delivering workable software to users as soon as possible, and incorporate their feedback into further development. 

**Fit for use**

The second major category of software quality is fitness for use. Software that is fit for use is capable of delivering its functionality to end users with sufficient availability, performance, and security. For end users, it needs to support the most common browsers. For system operators, it should provide clear log messages to support debugging in production. For disaster recovery purposes, it needs to support running in clustered modes of operation, even across data centers.

Ensuring fitness for use is the main driver for system architecture and the technical design of the software. During the definition phase, dedicated workshop tracks are devoted to determining the non-functional requirements. These can include requirements on how the software should conform to a target architecture, interface requirements for communicating with external systems, requirements about configuration and logging, and many others. Fitness for use is not only about technical requirements, however. An important requirement for most business is that software developed by Ximedes can potentially be maintained and further developed by third parties if and when the need arises. We take great care to make sure that this is possible by breaking down the software to small independent pieces. 

By combining only well-known and proven open source libraries and frameworks, we prevent any vendor lock-in, and no proprietary knowledge or tooling is required to build our software. And, as part of our standard documentation stack (based on the open J-STD-016 standard), everything a developer needs to know and do to setup a development environment is documented in a way that any professional software developer can easily follow.

**Availability is a joint effort**

Non-functional requirements, such as uptime guarantees and performance under loads, have become an increasingly important part of today’s business cases for cloud-based software. Achieving these demands requires close cooperation between infrastructure, sysadmins, and developers, to ensure seamless deployments and always on software services.
Just like functional testing, we believe that the non-functional tests validating that the software is fit for use should be automated and run as often as possible:

• Once (a part of) the software is sufficiently stable, a 24/7 burn-in test is developed, which puts our internal test environment under continuous load. This allows us to catch potential operational problems (such as degrading performance) early on.

• All source code is monitored on daily basis using the open source SonarQube software. SonarQube measures many different kinds of code quality metrics, such as code complexity, coding rule violations, unit test coverage, and evaluates the result against our internal benchmark. This ensures we deliver readable and maintainable code – an important factor in ensuring future maintainability.

• Often, non-functional requirements include the need to migrate data from an existing system to a new one. Again, we aim to test data migrations in the same automated way, migrating a representative source data set several times a day, and validating the results with specifically designed test cases.

Some non-functional test cases cannot be easily automated, and are validated in a production acceptance test (PAT) by the system operators. This include backup/re-store and disaster recovery tests, but also security penetration tests which should be performed at least annually in the actual production environment.

It is tempting to think that once software is shown to be fit for purpose and fit for use, all is well. However, if there is one certainty, it is that software needs to change many times during its lifecycle. That means that above all, to support your business, software needs to be fit for change.

TO MEASURE IS TO KNOW
When it comes to knowing how software behaves under operational conditions, no up-front design can beat an actual load- and performance test.

Fit for change

While often ignored, fitness for change is a crucial property for any software, be it off-the-shelf or bespoke, to be a viable choice for your business. The one constant in business is change – markets change, new rules and regulations make new demands on your processes, and before you know it you need to alter the way your software works.

One of the greatest benefits of using bespoke software products is that you can change it whenever and however you wish – there is no release calendar, no change approval by the vendor and no required license upgrades. Instead, Ximedes offers its Application Lifecycle Management services to guarantee your software can be changed, tested and delivered swiftly, against costs that are related only to the size of the change, and nothing else.

To be fit for change requires roughly three things:

• Well-designed, well-documented software, based on well-known technology, with clearly designed modules and readable source code

• A fully automated procedure to build, test and deploy the software in hours, not days or weeks

• The availability of senior software engineers who know the software and your business

In fact, many of the processes we touched upon earlier, such as our focus on automated testing and code quality monitoring, are an important part of being fit for change. If releasing a new version would involve a manual test effort of several weeks, even small changes would be prohibitive-ly costly. It is at these times that the investment in quality made during the initial development pays off, and pays off in a big way.
SIMPLE DESIGNS MINIMIZE THE COST OF CHANGE

Simplicity is the single most important aspect of good design. Engineers more easily understand simple software, which means they are productive sooner and make fewer mistakes. Simple software is easily tested automatically, which means releases are fast and fully tested, every time. And, simple software is easy to configure and deploy, which means your changes will be in production sooner.

Ximedes uses Kibana in combination with Logstash to search in log files and visualize the status. Simple graphs with colours indicate different log messages (error, info) that give a remarkable insight into the heartbeat of a healthy application and thus exceptional situations & errors can quickly be identified.

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Steps</th>
<th>Feature</th>
<th>Total</th>
<th>Passed</th>
<th>Failed</th>
<th>Total</th>
<th>Passed</th>
<th>Failed</th>
<th>Skipped</th>
<th>Pending</th>
<th>Duration</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Setting up the test context</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>22</td>
<td>32</td>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3 secs and 826 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Creating a new wallet</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>13</td>
<td>13</td>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>718 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Users can set a pincode</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>9</td>
<td>9</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100 ms</td>
<td>passed</td>
</tr>
<tr>
<td>The system can validate if users entered their correct pincode</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>15</td>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>160 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Users can add their cards to their wallets</td>
<td>28</td>
<td>28</td>
<td>0</td>
<td>150</td>
<td>150</td>
<td>0</td>
<td>150</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 min and 54 secs and 969 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Provisioned Cards can be inserted via Soap</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>156 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Users can add cards from their wallets</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>14</td>
<td>14</td>
<td>0</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>29 secs and 281 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Initialize Checkout</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>144 ms</td>
<td>passed</td>
</tr>
<tr>
<td>When an user wants to checkout, a call is made to the masterpass api</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>632 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Users can delete their cards from their wallets</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>40 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Delete a wallet</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Validate shipping address</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>114 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Initialize checkout</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>12</td>
<td>12</td>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>144 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Delete a wallet</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>11</td>
<td>11</td>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>91 ms</td>
<td>passed</td>
</tr>
<tr>
<td>Validation shipping address</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>114 ms</td>
<td>passed</td>
</tr>
<tr>
<td>12</td>
<td>80</td>
<td>80</td>
<td>0</td>
<td>289</td>
<td>289</td>
<td>0</td>
<td>289</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2 mins and 29 secs and 241 ms</td>
<td>Total</td>
</tr>
</tbody>
</table>

Image 7 - Fast, repeatable regression test with every release

Image 8 - A Kibana dashboard shows real-time aggregated logging data

ASK OUR CTO EVERYTHING YOU WANT TO KNOW ABOUT SOFTWARE DEVELOPMENT.

Dr. Joris Portegies Zwart - CTO
T: 31 88 - 24 81 632
E: joris.portegies.zwart@ximedes.com