Techniques
1. How we work
2. Unit Testing
   a. Test Driven Development
3. Toolchain (TFS)
   a. Work Item Tracking
   b. DevOps
   c. Code Quality
4. Frameworks
5. Core Technologies
How we work
What we can provide - Teams

- K&K “A-Team”: 5 full-time developers with deep specialisation in C#, XAML (WPF) & T-SQL (Microsoft SQL Server)
  - For peak throughput, support by our other K&K “Team X” possible (also 5 C# developers)
  - K&K can thus provide up to 1500 hours of development per month
- We can accommodate external developers and integrate them into the project
- We can also contract an UI/UX expert to support your wishes
Industry supported best practice

- Coding Conventions and Guidelines from Microsoft
  - Coding Conventions
  - Guidelines:
    - Framework Design Guidelines
      https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines
    - User-Experience
      https://docs.microsoft.com/en-us/windows/desktop/uxguide/guidelines
  - Language Features (new in version 7.3)
    - https://docs.microsoft.com/en-us/dotnet/csharp/whats-new/csharp-7-3
Unit Testing & Test Driven Development
To ensure the quality of code, almost every component has to be runnable without the normal application context.

Must be implemented from conception, otherwise there is a risk that individual components can no longer be separated from the rest of the application and thus tested.

Unit testing as basis for proper and lasting software quality:

- Provides the basis for Test Driven Development (see next slide).
- Generally Test Driven Development is more expensive initially, but saves resources in the long term. Even when pure Test Driven Development is unfeasible, many elements of this method of development can be adapted to fit “normal” projects.
Unit Testing: Test Driven Development

- This method of development yields earliest possible error detection and avoidance.
- Focus work on the functionality and the primary features of the software.
- Acceptance criteria and dependencies are turned into unit tests, piece by piece.
- Every change is fully automatically tested. We can recognize and correct errors while we implement user stories.
Toolchain

- Visual Studio (opt. Productivity Power Tools)
- Sql Server Management Studio (SSMS)
- Team Foundation Server (TFS)
  - Transparent communication and documentation at a glance
    - Shows which acceptance criteria meets the implemented user story
    - Shows all code changes to meet individual acceptance criteria
    - Shows who implemented each user story
  - Automated builds and release management
    - Is provided by us
    - Every version can be compiled and downloaded seamlessly
  - Accessible via the internet
<table>
<thead>
<tr>
<th>Order</th>
<th>Work Item Type</th>
<th>Title</th>
<th>State</th>
<th>Effort</th>
<th>B...</th>
<th>Value Area</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>Epic</td>
<td>Installation</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feature</td>
<td>Installationspaket</td>
<td>Done</td>
<td>5</td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feature</td>
<td>Datenverzeichnis der Userdaten</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product Backlog</td>
<td>Installation von SQL Server stand-alone</td>
<td>Done</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task</td>
<td>Konfigurieren des Setup Projekts</td>
<td>Done</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product Backlog</td>
<td>Installation von SQL Server</td>
<td>Done</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task</td>
<td>Setup Projekt konfigurieren</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task</td>
<td>Testen unter unterschiedlichen Systemen</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task</td>
<td>instellen bauen</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Epic</td>
<td>Installationspfad für SQL Server</td>
<td>Done</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Epic</td>
<td>newadmin</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Epic</td>
<td>newweb</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Epic</td>
<td>Architektur</td>
<td>In Prog...</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feature</td>
<td>Datenbank</td>
<td>In Prog...</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product Backlog</td>
<td>Als Entwickler möchte ich eine normalisierte D...</td>
<td>Done</td>
<td>3</td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task</td>
<td>Konzeption der Datenstruktur in einem Team-Meeting / Workshop</td>
<td>Done</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task</td>
<td>Erstellen der Datenbank anhand des erarbeiteten Konzepts</td>
<td>Done</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Task</td>
<td>EF einbinden &amp; erste Modelle erstellen</td>
<td>Done</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product Backlog</td>
<td>Als Nutzer möchte ich einen Datenbank-...</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feature</td>
<td>Altdatenimport</td>
<td>In Prog...</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product Backlog</td>
<td>Als Administrator möchte ich Altdaten aus Excel in die Datenstruktur importieren können, damit ich diese weiterverwenden ...</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Feature</td>
<td>Webanwendung</td>
<td>New</td>
<td></td>
<td></td>
<td>Business</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Product Backlog</td>
<td>Als Entwickler möchte ich eine Grundstruktur des Softwareprojekts vorfinden, damit ich meine Features dort implementiere...</td>
<td>Done</td>
<td>3</td>
<td></td>
<td>Business</td>
<td></td>
</tr>
</tbody>
</table>
One unit test per acceptance criteria
Changeset 5386 - fixed bug: crash when creating a new display...

Leiber, Thomas (Mar 18 04:27 (UTC +01:00) - 0 minutes ago)

Associated work items:
- [10014] AudiScreen registers the Mobile on Client
- [8003] Verhalte des Screen-Klickverhaltens anpassen

Associated files:
- ScreenManagementService.cs
- ScreenManagementService.cs

C#:

```csharp
if (displayClient != null)
    this.closeWindows(new DisplayClient(location, null, location.ID));
else if (displayClient != null)
    this.closeWindows(new DisplayClient(location, null, location.ID));

if (displayClient != null)
    this.closeWindows(new DisplayClient(location, null, location.ID));
```

- Rauszyk, Folko yesterday
  - Guter Eintrag, Thomas! Dass hier asynchon gearbeitet wird, muss der Code überall aushalten werden.

- Leiber, Thomas a hour ago
  - Einer für den Lärm, Rauszyk, Folko!
  - Ich habe bereits alle Anfragen mit dem zweiten Parameter ausgeführt, weiterhin
  - Bug 10303: Displaysoftware-crashed wenn ein externer PC herangezogen wird

Modekwein-Software crashed wenn ein externer PC herangezogen wird...
Test Results

Abgeschlossene Ausführungen

Tests gesamt: 307 (+307)
- Erfolgreich (307)
- Fehler (0)
- Andere (0)

Tests mit Fehlern: 0 (+0)
- Neu (0)
- Vorhanden (0)

Durchlaufprozentsatz: 100% (+100%)

Testlaufdauer: 58 s 500 ms (+58 s 500 ms)

Detaillierter Bericht >

Code Coverage

release any cpu

Blocks: 56.90% 7.017/12.333

Lines: 57.87% 3.862/6.674

Code Coverage-Ergebnisse herunterladen

Tags

Add tag...

Deployments

Für diesen Build wurden keine Bereitstellungen gefunden. Release erstellen.
**Quality Gate**  Passed

Some Quality Gate conditions on New Code were ignored because of the small number of New Lines 🔄

<table>
<thead>
<tr>
<th>Bugs</th>
<th>Vulnerabilities</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 🟢</td>
<td>11 🟢</td>
</tr>
<tr>
<td>New Bugs</td>
<td>New Vulnerabilities</td>
</tr>
<tr>
<td>0 🟢</td>
<td>0 🟢</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Code Smells</th>
</tr>
</thead>
<tbody>
<tr>
<td>2d 🟢</td>
</tr>
<tr>
<td>New Debt</td>
</tr>
<tr>
<td>0 🟢</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>36.5%</td>
</tr>
<tr>
<td>Coverage</td>
</tr>
<tr>
<td>0.0%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Duplications</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.5%</td>
</tr>
<tr>
<td>Duplications</td>
</tr>
<tr>
<td>0.0%</td>
</tr>
</tbody>
</table>

Leak Period: since previous version started 5 months ago

Project Activity

- September 27, 2018
  - 1.0
- May 16, 2018
  - Quality Profile: Use 'A-Team Rulez' (C#)
  - Quality Profile: Stop using 'Sonar way' (C#)
- May 11, 2018
  - Project Analyzed
  - Show More

Quality Gate

(Default) Sonar way

Quality Profiles

(C#) A-Team Rulez

Project Key

[Copy]
Frameworks and Techniques
Current frameworks and Techniques

Use of best practice guidelines ⇒ we do not reinvent the wheel.

- PRISM Framework
- Repository in conjunction with Unit-of-Work design pattern
- Dependency Injection
- Façades
- Unit Testing
- Entity Framework
● PRISM is a powerful open source MVVM library

● Glossary
  ○ **MVVM** = loose coupling between Model, View and ViewModel
  ○ **Model** = data model (intermediate layer to represent the data from the database)
  ○ **View** = display objects (graphical user interface & visual representation of ViewModel)
  ○ **ViewModel** = state of the interface (and link between model and view)

● Benefits
  ○ Supporting design patterns that embody important principles of architectural design (e.g. “Separation of Concerns”), PRISM helps developers design and build loosely-coupled components that can be developed independently, but jet can be easily and seamlessly integrated into the overall application.
Repository and Unit-of-Work pattern

● Repository pattern in conjunction with unit-of-work pattern
  ○ Glossary
    ■ Repository ⇒ Representation and collection of database operations for business logic
    ■ Unit-of-Work ⇒ Collection of repositories within a mask
  ○ Benefits
    ■ Data is created, edited, and deleted centrally in one place
    ■ Prevention of duplicate code as this layer can be accessed from anywhere

● ⇒ We are writing future-oriented and maintainable code!
Dependency Injection

- Is a technique to minimize fixed references
- Increases the testability due to hiding complex code fragments behind simplified interfaces => code lines are not needlessly tested twice
- The library “Unity” is used as our Dependency Injector
- Centralized dependency management, which “inserts” the appropriate code fragment (e.g. Service, Factory, or Façade, etc.) into objects
Façade

- Hides complex code fragments behind a simple “facade”
  - Code irrelevant to the developer is hidden
    ⇒ improves readability of code
    ⇒ especially important if the code is modified by several developers
Frontend frameworks

- Xceed Toolkit
  - See Live-Explorer for Demo: [https://github.com/xceedsoftware/wpftoolkit/releases](https://github.com/xceedsoftware/wpftoolkit/releases)
  - AvalonDock
    - Window management for complex applications
  - User Controls
    - A variety of components for editing, creating, and deleting data
    - Easily expandable
Core Technologies
C# common toolsets

- Nuget Package Manager to provide specially designed source code libraries
- Encryption and authentication of communications to ensure data integrity and security
- High-performance applications with multi-threaded programming

- “MVP” Concept
  - Minimum Viable Product
- Design patterns
  - Dependency Injection (Unity, …)
  - Asynchronous Programming Model
  - Inversion of Control
  - ...
- We support a wide range of database management systems. Depending on project requirements Microsoft SQL Server or SQLite are preferred.
- Unit Testing
  - NUnit
  - Moq
Frontend Technology

Desktop-Client

- Windows Presentation Foundation *(WPF)*
- MVVM Light Toolkit or PRISM, depending on the scope of the project.
- OxyPlot to generate graphs and diagrams
- Xceed Toolkit to support your UI needs.
- AvalonDock to support multi window applications

Web-Client

- Backend
  - ASP.NET MVC 6 or ASP.NET Core MVC depending on platform
- Frontend
  - Current web html5 technology
  - Typescript
  - Bootstrap 4
  - Angular
Any questions? Please contact us!

We are committed to your request and will be happy to answer detailed questions promptly. We look forward to your inquiry.

I am your contact person:

Laura Köpl
Head of Marketing and Sales
Phone: 09382 / 3102-241
Mail: koepl@kk-software.de
Protect your interests:

Develop excellent products with our team at K&K Software AG using state-of-the-art agile methods, now.