SEMINAR AUS SOFTWARE-ENTWICKLUNG

SCRUM2

Abschlussbericht

Alexander Duml
Stefan Müller
Thomas Perl
Martin Wieser

Inhaltsverzeichnis

Prozessbeschreibung

Scrum ist ein schlanker, agiler Managementprozess, welcher Anfang der 1990er Jahre von Ken Schwaber und Jeff Sutherland für die Umsetzung komplexer Produktentwicklungen konzipiert wurde und heute insbesondere in der Software-Entwicklung Anwendung findet. Kennzeichnend für Scrum ist unter anderem das inkrementelle, interaktive und empirische Vorgehen.

Idealtypischer Ablauf

Abbildung: Scrum Ablaufschema[1]

Das Ziel von Scrum ist es eine Produkt-Vision schnell, kostengünstig und qualitativ hochwertig umzusetzen. Dafür werden zunächst in einer strategischen Planungsphase, die aus der Vison abgeleiteten Produktfunktionalitäten, im sogenannten Product Backlog festgehalten und entsprechend ihres Geschäftswerts priorisiert.

Anschließend beginnt die eigentliche Umsetzung, wobei diese in einer Serie von Sprints voranschreitet. Ein Sprint ist dabei als eine immer gleichlange, typischerweise zwei bis vier Wochen dauernde Zeitspanne zu verstehen. Zu Beginn jedes Sprints steht das Sprint Planning Meeting. In diesem Meeting wählt das Entwicklungs-Team die am höchsten priorisierten Items aus dem Product Backlog aus und definiert, wie diese während des Sprints umgesetzt werden sollen. Das resultierende Artefakt wird als Sprint Backlog bezeichnet.

Die im Sprint Backlog definierten Items werden während des Sprints entworfend, programmiert und getestet, sodass am Ende des Sprint ein auslieferbares Stück Software steht. Idealerweise trifft sich das Entwicklungs-Team während eines Sprints an jedem Arbeitstag zum sogenannten Daily Scrum, einem ca. 15-minütigen Meeting, welches u.a. der Planung des nächsten Tages dient. Bis zum Projektabschluss werden mehrere Sprints durchgeführt, sodass das Produkt kontinuierlich weiterentwickelt wird.

Das Scrum-Framework wird durch wenige, dafür aber sehr effektive Rollen, Regeln, Meetings und Artefakte determiniert. Im Folgenden werden diese kurz erörtet.

Rollen

Scrum sieht die folgenden Rollen vor:

[1] Quelle: www.scrumalliance.org

- **Der Product Owner:** Zu den Aufgaben des Product Owners gehört es die Bedürfnisse der Kunden und Stakeholder zu erfassen und diese im Product Backlog, welches er verwaltet, abzubilden. Er trägt die Verantwortung für das Produkt und den Projekterfolgt, legt die Prioritäten fest, hat sicherzustellern, dass das Entwicklungsteam die einzelnen Product Backlog-Einträge versteht und akzeptiert oder weist Arbeitsergebnisse zurück. Scrum sieht dabei ausdrücklich vor, dass der Product Owner eine Person und kein Ausschuss oder Gremium ist. Außerdem ist der Product Owner kein Mitglied des Entwicklungs-Teams.

- **Der Scrum Master:** In der Verantwortung des Scrum Master liegt es, dass Scrum verstanden und richtig umgesetzt wird. Dazu führt er gegebenenfalls Coaching der Teammitglieder durch oder versucht den Lernprozess des Teams anzustoßen. Er moderiert die diversen Meetings und bereitet diese vor. Zudem schützt der Scrum Master das Team vor äußeren Störungen und beseitigt etwaige Hindernisse. Der Scrum Master ist kein Teil des Entwicklungsteams. Er arbeitet weder inhaltlich, noch hat er eine Weisungsbefugnis gegenüber dem Team.

- **Das Entwicklungs-Team:** Das Team besteht üblicherweise aus fünf bis neun Mitglieder. Es ist interdisziplinär besetzt und verfügt über das gesamte Wissen und Können um das Produkt zu realisieren. Die Teammitglieder verpflichten sich das selbst definierte Sprint-Ziel zu erreichen. Dabei arbeiten sie selbstorganisiertes, was bedeutet, dass niemand vorschreibt wie die Einträge des Product Backlogs konkret umzusetzten sind. Auch werden keine Hierarchien von außen diktiert.

Zusätzlich sieht Scrum weitere Rollen, die zwar nicht Teil des eigentlichen Prozesses sind aber dennoch auf diesen einwirken, vor. Darunter fallen beispielsweise die Rollen User, Customer oder das höhere Management.

Meetings

Scrum sieht die folgenden Meetings vor:

- **Sprint Planning:** Das Sprint Planning Meeting steht zu Beginn eines jeden Sprints und gliedert sich in zwei Teile:

    - Im ersten Teil erfolgt die Auswahl der zu bearbeitenden Product Backlog-Einträge. Dazu stellt der Product Owner dem Team die geordneten Einträge vor, um ein gemeinesames Verständnis zu erhalten. Die Anzahl der im Sprint umzusetzenden Items wird dann vom Team bestimmt und richtet sich danach, ob die Umsetzung in der fest vorgegebenen Sprintdauer erfolgen kann. In Folge verpflichtet sich das Team das Sprint-Ziel zu erreichen.

    - Im zweiten Teil des Meetings entscheidet das Team, wie genau die ausgewählten Einträge umgesetzt werden. Das Team definiert alle dafür notwenigen Aktivitäten und es erfolgen erste Diskussionen über Architektur, Design etc. Das resultierende Artefakt wird als Sprint Backlog bezeichnet.

- **Daily Scrum:** Dieses Meeting dauert 15 Minuten und findet täglich zur selben Zeit am selben Ort statt. In der Regel nehmen Scrum Master und Entwicklungs-Team daran teil. Ziel ist die Synchronisation der Aktivitäten, der Informationsaustausch und das Planen der nächsten 24 Stunden. Es werden Hindernisse diskutiert und durch Scrum-Master in der Impediment List erfasst.

- **Sprint Review:** Am Ende jedes Sprints steht das Sprint Review Meeting. Es dient der Präsentation der Ergebnisse durch das Team, der Überprüfung der Zielerreichung, der Abnahme der Arbeitsfortschritte durch den Product Owner und der Reflektion. Je nach Sprintlänge dauert es zwischen zwei und vier Stunden.

- **Sprint Retrospective:** Dieses Meeting findet ebenfalls am Ende des Sprints statt und dauert ca. 2-3 Stunden. Es dient dazu, die Zusammenarbeit des Teams zu verbessern, die Produktivität und Softwarequalität zu steigern und die Anwendung des Scrum-Prozesses zu optimieren. Der Scrum Master steht dem Team dabei unterstützend zur Seite und die erarbeiteten Verbesserungen werden im nächsten Sprint umgesetzt.

Artefakte

Im Zentrum von Scrum stehen die folgenden Artefakte:

- **Product Backlog:** Das Product Backlog ist eine Liste aller Anforderungen an die Software. Das Artefakt liegt in der Verantwortung des Product Owner. Dieser erfasst bei Projektstart alle Anforderungen auf Basis der Produkt-Vision und während des Projekts führt er eine fortlaufende Detaillierung und Aktualisierung durch. Im Product Backlog erfolgt eine priorisierung nach Geschäftswert. Das Dokument ist von allen einsehbar und jeder kann Einträge einbringen.

- **Sprint Backlog:** Das Sprint Backlog ist eine Liste aller im laufenden Sprint durchzuführenden Aktivitäten inklusive einer Schätzung. Es wird durch das Entwicklungs-Team während des Sprint Plannings erstellt und liegt in dessen Verantwortung.

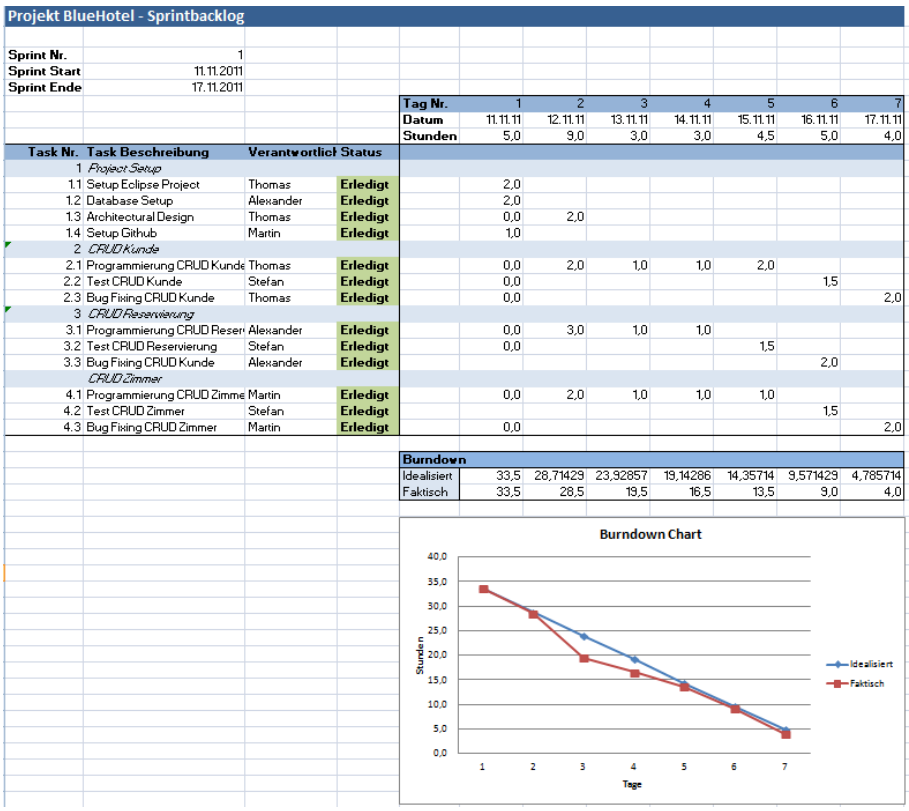
Weitere von Scrum vorgesehene Artefakte sind: Impediment List (Liste von Hindernissen), Definition of Done (Definition des Verständnis von „fertig“), Produktvision und Burndown Charts (grafische Darstellung des Sprint-Fortschritts).

Technologien

Folgende Technologien wurden von uns verwendet:

Excel

Excel verwendeten wir zunächst für die Umsetzung von Sprint Backlog und Burndown Chart. Die folgende Abbildung zeigt die Artefakte für den ersten Sprint:



Projekt BlueHotel - Sprintbacklog

Sprint Nr.	1
Sprint Start	11.11.2011
Sprint Ende	17.11.2011

Task Nr.	Task Beschreibung	Verantwortlich	Status	Tag Nr.	1	2	3	4	5	6	7
				Datum	11.11.11	12.11.11	13.11.11	14.11.11	15.11.11	16.11.11	17.11.11
				Stunden	5,0	9,0	3,0	3,0	4,5	5,0	4,0
1	*Project Setup*										
1.1	Setup Eclipse Project	Thomas	Erledigt		2,0						
1.2	Database Setup	Alexander	Erledigt		2,0						
1.3	Architectural Design	Thomas	Erledigt		0,0	2,0					
1.4	Setup Github	Martin	Erledigt		1,0						
2	*CRUD Kunde*										
2.1	Programmierung CRUD Kunde	Thomas	Erledigt		0,0	2,0	1,0	1,0	2,0		
2.2	Test CRUD Kunde	Stefan	Erledigt		0,0					1,5	
2.3	Bug Fixing CRUD Kunde	Thomas	Erledigt		0,0						2,0
3	*CRUD Reservierung*										
3.1	Programmierung CRUD Reser	Alexander	Erledigt		0,0	3,0	1,0	1,0			
3.2	Test CRUD Reservierung	Stefan	Erledigt		0,0				1,5		
3.3	Bug Fixing CRUD Kunde	Alexander	Erledigt							2,0	
	CRUD Zimmer										
4.1	Programmierung CRUD Zimme	Martin	Erledigt		0,0	2,0	1,0	1,0	1,0		
4.2	Test CRUD Zimmer	Stefan	Erledigt							1,5	
4.3	Bug Fixing CRUD Zimmer	Martin	Erledigt		0,0						2,0

Burndown							
Idealisiert	33,5	28,71429	23,92857	19,14286	14,35714	9,571429	4,785714
Faktisch	33,5	28,5	19,5	16,5	13,5	9,0	4,0

Abbildung: Sprint Backlog und Burndown für Sprint 1

Nach den ersten beiden Sprints hat sich gezeigt, dass das regelmäßige Aktualisieren des Dokuments nur sehr nachlässig durchgeführt wurde. In der Retrospective zu Sprint 2 stellt sich heraus, dass der Gund dafür war, dass der Vorgang als zu umständlich erachtet wurde (Pull vom Repository, Dokument öffnen und anpassen, Dokument pushen). Wir haben uns infolge dessen nach einer anderen Lösung umgeschaut und uns schlussendlich für eine Abbildung von Sprint Backlog und Burndown mittels GitHub-Issues entschieden.

Im weiteren Projektverlauf wurde Excel ausschließlich dazu verwendet die Testfälle zu verwalten.

GitHub

GitHub stellte die wichtigte Technologie in unserem Projekt dar. Von großer Bedeutung war insbesondere die Issue-Liste. Diese nutzen wir u.a. um das Sprint Backlog abzubilden. Das Vorgehen gestaltete sich dabei wie folgt:

- Während des Sprint Planning Meetings wurden für jeden umzusetzenden Product Backlog-Eintrag alle dazu nötigen Tasks als Issue in die Liste eingetragen.
- Die entsprechenden Issues wurden mit dem Lable „TODO“ versehen und den verantwortlichen Personen zugewiesen.
- Zusätzlich wurden die Issues mit dem Milestone „End of Sprint X“ verknüpft.

Folgend ist als Beispiel die Issue-Liste von Sprint 6 angeführt:

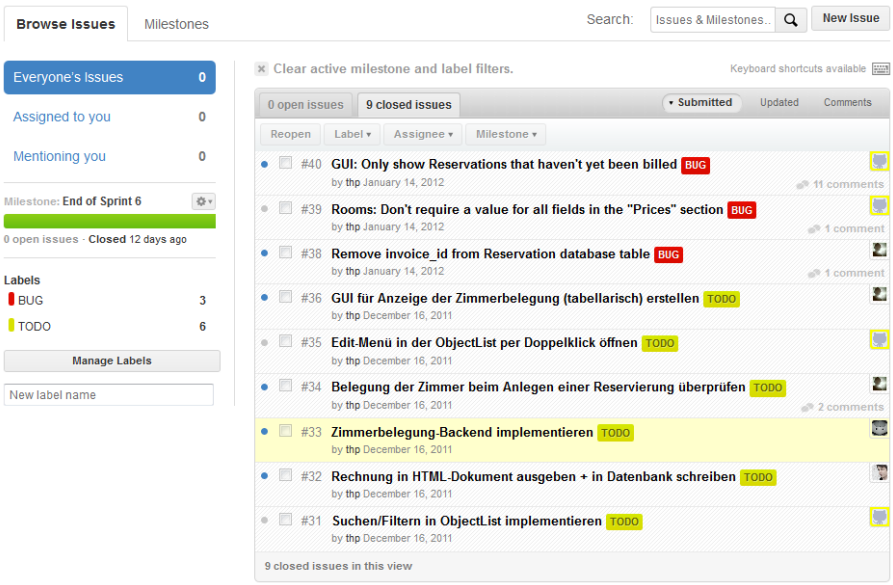


Abbildung: Issue-Liste zu Sprint 6

Da das Updaten der Issues online im Browser vorgenommen werden konnte bzw. die Issues auch automatisch beim Commit des Sourcecodes geschlossen wurden, war das Commitment im Vergleich zur vorhergehenden Excel-Lösung viel höher.

In die Issue-Liste wurden ebenfalls die Bugs verwaltet, wobei diese mit dem Label „BUG“ gekennzeichnet wurden. Durch die konsequente Verwendung der Milestones ließ sich ebenfalls eine Art Burndown Chart realisieren. Die folgende Abbildung zeigt dazu eine Übersicht:

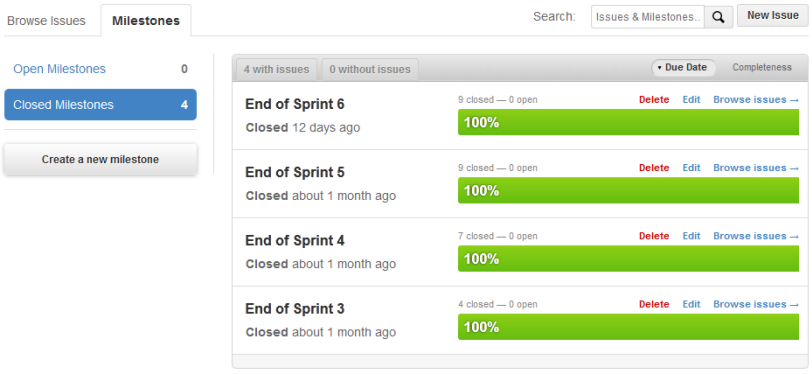


Abbildung: „Burndown Charts“ zu den Sprints 3-6

Schlussendlich wurde mittels GitHub-Wiki auch das Product Backlog abgebildet. Dadurch hatte alle schnell und einfach einen Einblick in dieses Dokument.

E-Mail

Die E-Mail-Kommunikation wurde hauptsächlich für organisatorisches verwendet – beispielsweise das Organisieren des wöchentlichen Meetings.

Handy

In Ausnahmefällen wurde kurzfristig auch mittels Handy kommuniziert.

Anpassungen

Folgen werden die einzelnen Prozess-Anpassungen kurz erörtert. Eine detaillierte Beschreibungen der Auswirkungen kann dem Kapitel Retrospektive entnommen werden.

Anpassung 1 – Verzicht auf Daily Scrum

Das Daily Scrum wurde von uns nicht durchgeführt. Ein Grund dafür ist, dass es sich im studentischen Umfeld als äußerst schwierig gestaltet, sich jeden Tag zur selben Zeit am selben Ort zu treffen. Zudem wurde von uns der Nutzen eines solchen Meetings als gering eingeschätzt. Die eigentlich im Daily durchgeführten Aktivitäten (Informationsausstausch, Synchronisation, Beseitigen von Hindernissen) wurde, falls erforderlich, über E-Mail-Kommunikation bzw. GitHub-Issues durchgeführt.

Anpassung 2 – Rollen Kunde, Benutzer und Management werden vernachlässigt

In unserem Projekt gestaltete sich die Rollenverteilung wie folgt: Martin Wieser als Product Owner, Stefan Müller als Scrum Master und Thomas Perl sowie Alexander Duml als Entwicklungs-Team.

Alle weiteren Rollen wurden von uns nicht berücksichtig. Da diese – wie oben beschrieben – nicht Teil des eigentlichen Prozesses sind, sahen wird dafür keinen Bedarf.

Anpassung 3 – Jeder ist Entwickler

Entgegen der Scrum-Vorgaben arbeiteten sowohl der Product Owner als auch der Scrum Master im Entwicklungs-Team mit. Uns ist sehr wohl bewusst, dass dies in einem realen Projekt zu Konflikten führen würde und dieses Vorgehen daher nicht akzeptabel wäre. In unserem Projekt erachteten wir dies allerdings als vernachlässigbar.

Zudem gingen wir davon aus, dass Product Owner und Scrum Master eine geringere Arbeitslast zu tragen haben werden als das Entwicklungs-Team. Im Sinne einer faireren Verteilung der Arbeitslast entschieden wir uns daher für die beschriebene Lösung.

Anpassung 4 – Kürzere Sprintdauer als vorgeschlagen

Wir haben uns entgegen der Empfehlung von Scrum (2-4 Wochen) für eine Sprintdauer von lediglich einer Woche entschieden. Die Idee war, dass wir kleine, dafür aber relative viele Sprints durchführen wollten, um damit in Verbindung mit einem wöchentlichen Meeting eine bessere Kontrolle über den Projektfortschritt zu gewährleisten.

Anpassung 5 – Ein großes Meeting

Wir haben uns dafür entschieden, ein regelmäßiges Meeting ein mal pro Woche im Umfang von ein bis zwei Stunden abzuhalten (typischerweise Freitags am selben Ort und zur selben Zeit). In diesem Meeting wurde zunächst Sprint Review und Sprint Retrospective für den vergangenen Sprint durchgeführt, um anschließend mit dem Sprint Planning des folgenden Sprints abzuschließen. Aufgrund der kurzen Sprintdauer waren auch die Meetings entsprechend kürzer.

Die Scrum-Empfehlungen hinsichtlich Meeting-Dauer wurde somit unterschritten.

Anpassung 6 – Optimierung der Artefakte

Auf die Produktvision wurde ebenfalls verzichtet, wobei dies im Charakter des Projekts begründet liegt (Produkt mehr oder weniger Vorgegeben).

Projektverlauf

04.11.2011 – Kickoff-Meeting

Beim Kickoff Meeting wurden die Rollen der Teammitglieder verteilt und es wurde eine Technologieauswahl getroffen:

- Repository auf GitHub
- Dokumentation in LaTeX
- Programmiert wird in Java
- GUI mit Swing realisiert
- HSQLDB als Datenbank

Weiters sind die nächsten Schritte definiert worden:

- Sich mit SCRUM vertraut machen
- LaTeX Template für die Dokumentation erstellen
- User Stories zum Product Backlog hinzufügen
- GitHub einrichten
- Projekt einrichten (Architektur definieren)

11.11.2011 – Zweites Meeting

Review

Das Projekt wurde erfolgreich aufgesetzt und eine Architektur wurde definiert. Außerdem wurden für die Softwareentwicklung zwei neue Technologien eingesetzt, welche vorher noch nicht spezifiziert worden sind:

- Eclipse als Entwicklungsumgebung
- Hibernate als OR-Mapper in Kombination mit JPA Annotations in den Models

Retrospektive

Da es sehr anstrengend ist Swing GUIs mit Hand zu implementieren soll das Windows Builder Eclipse Plugin genutzt werden.

Planning Meeting 1

Es sollen folgende User Stories implementiert werden:

- Reservierung vornehmen:
  Als Rezeptionist will ich schnell und einfach neue Reservierungen anlegen, so dass ich mich wieder den Kunden widmen kann.

- Kunde anlegen:
  Als Rezeptionist will ich neue Kunden erstellen können, so dass ich ihre Daten für die Rechnung habe.
- Zimmer anlegen:
  Als Geschäftsführerer will ich neue Zimmer hinzufügen, so dass das System nach einen Hotelausbau korrekt läuft.
- Daten löschen:
  Als Geschäftsführerer will ich alte Daten löschen können, so dass das System nach einen Hotelumbau korrekt läuft.
- Daten bearbeiten:
  Als Geschäftsführerer will ich Daten ändern können, so dass fehlerhafte Eingabe korrigiert oder Informationen ergänzt werden können.

Planning Meeting 2

Die User Stories werden folgendermaßen implementiert:

- CRUD (Create, Read, Update und Delete) Mechanismus für Kunde (Model für die Datenbankanbindung + Logik für Berechnungen + GUI für die grafische Darstellung der Daten)
- CRUD Mechanismus für Zimmer (Model + Logik + GUI)
- CRUD Mechanismus für Reservierung (Model + Logik + GUI)

18.11.2011 - Sprint 1

Review

Alle geplanten Features wurden implementiert, aber es wurden keine Testfälle spezifiziert und somit wurde nichts getestet. Deshalb werden die Features auf den nächsten Sprint verschoben.

Außerdem wurde vom Product-Owner bemängelt, dass die Ansicht von Kunden-, Zimmer- und Reservierungsliste in einzelnen Fenstern geöffnet wird, es sollte alles in einem Fenster gehalten werden.

Das Entwicklerteam findet die Implementierung der GUI von Reservierung ist unsauber, die Berechnungen sollen in eine Logik-Komponente ausgelagert werden.

Retrospektive

Da bis jetzt keine Testfälle spezifiziert waren, müssen unbedingt Unit-Tests und funktionale Tests erstellt werden. Die Unit-Tests soll die Logik-Komponente auf korrekte Funktionalität prüfen. Die GUI soll funktionalen Tests unterzogen werden. Gefundene Fehler werden in die Bug-Liste (Excel Tabelle) eingetragen.

Planning Meeting 1

Es sollen folgende User Stories implementiert werden:

- Reservierung mit mehreren Zimmern:
  Als Rezeptionist will ich Reservierungen mit mehreren Zimmern anlegen, so dass bei einer Stornierung alle Zimmer frei werden.

- Reservierung mit mehreren Kunden:
  Als Rezeptionist will ich Reservierungen mit mehreren Kunden anlegen, so dass zukünftige Discounts korrekt berechnet werden.

Planning Meeting 2

Die User Stories werden folgendermaßen implementiert:

- CRUD Mechanismus für Reservierung (Model + Logik + GUI) aktualisieren, damit mit einer Reservierung mehrere Zimmer von mehreren Kunden gebucht werden können.

Die Verbesserungsvorschläge, welche aus Review und Retrospektive hervorgehen, werden folgendermaßen erledigt:

- Logik aus GUI in eine Logik-Komponente auslagern
- Unittests für Logik erstellen
- GUI Refactoring
- Template für Testfälle erstellen

25.11.2011 - Sprint 2

Review

Alle geplanten Features von Sprint 1 und 2 wurden implementiert und erfolgreich getestet. Auch der Product-Owner war mit dem überarbeiteten GUI zufrieden.

Retrospektive

Es gab im aktuellen Sprint keine Probleme im Prozess und deshalb auch keine Verbesserungsvorschläge.

Planning Meeting 1

Es sollen folgende User Stories implementiert werden:

- Kundendaten einsehen:
  Als Geschäftsführer will ich Einblick in die Kundendaten, so dass ich mit ihnen Kontakt aufnehmen kann.
- Rechnung erstellen:
  Als Rezeptionist will ich Rechnungen erstellen, so dass ich diese den Kunden vorlegen kann.

Planning Meeting 2

Die User Stories werden folgendermaßen implementiert:

- Aktualisierung der Kundenansicht
- CRUD Mechanismus für Rechnung (Model + Logik + GUI)

02.12.2011 - Sprint 3

Review

Alle geplanten Features wurden implementiert und erfolgreich getestet.

Retrospektive

Bug-Liste ist im Excel-Dokument schwer zum tracken. Deshalb soll die Issue-Liste von GitHub verwendet werden. Somit können die Fehler den zuständigen zugewiesen werden. Außerdem sollen Fehler und Features den jeweiligen Sprints durch GitHub-Milestones zugeordnet werden.

Der Product Backlog soll auch ins GitHub-Wiki verschoben werden.

Planning Meeting 1

Es sollen folgende User Stories implementiert werden:

- Reservierung stornieren:
  Als Rezeptionist will ich Reservierungen stornieren.
- Frühzeitige Abreise erfassen:
  Als Rezeptionist will ich ein frühzeitige Abreise erfassen, so dass die Zimmer wieder als "frei" erkannt werden.
- Rechnungen anzeigen:
  Als Geschäftsführer möchte ich mir schnell und einfach alle ausgestellten und noch offenen Rechnungen anzeigen lassen.

Planning Meeting 2

Die User Stories werden folgendermaßen implementiert:

- Hinzufügen eines Strono-Flags im Model, Erstellen eines Strorno-Buttons in der Reservierungs-Listen GUI und Implementieren der Storno-Logik
- Überarbeiten der Rechnungs GUI, hinzufügen der Logik für die frühzeitige Abreise und hinzufügen einer Liste mit allen Rechnungen

09.12.2011 - Sprint 4

Review

Alle geplanten Features wurden implementiert. Bei den funktionalen Tests wurde festgestellt, dass die ManyToMany Beziehung nicht ordnungsgemäß funktioniert. Dies hatte aber keine Auswirkung auf die derzeitige Funktionalität.

Der Benutzer findet, dass die Berechnung des Preises der Reservierung automatisch erfolgen sollte.

Außerdem sollte es beim Löschen eines Datensatzes eine Rückfrage geben bevor dieser gelöscht wird.

Retrospektive

Es gab im aktuellen Sprint keine Probleme im Prozess und deshalb auch keine Verbesserungsvorschläge.

Planning Meeting 1

Es sollen folgende User Stories implementiert werden:

- Automatische Berechnung des Preises bei Reservierung:
  Als Rezeptionist will ich dass sich der Preis beim Erstellen einer Reservierung automatisch berechnet.
- Löschung mit Rückfrage:
  Als Rezeptionist will ich, dass beim Löschen eines Datensatzes noch einmal nachgefragt wird ob ich den Datensatz wirklich löschen will.

Planning Meeting 2

Die User Stories werden folgendermaßen implementiert:

- Es wird eine Berechnung in die Logik-Komponente eingefügt, welche vom GUI durch ein Event aufgerufen wird.
- Der Lösch-Button der Kunden-, Zimmer- und Reservierungslisten GUI erhält eine Rückfrage, die bestätigt werden muss.

Bug-Fix von:

- Reservierungs-Model ManyToMany Annotation zu Kunde.

16.12.2011 - Sprint 5

Review

Alle geplanten Features wurden implementiert. Durch die funktionalen Tests wurde erkannt, dass Kunden, Zimmer und Reservierungen erstellt werden können. Es sollte jedem Editor eine Eingabe-Validierung hinzugefügt werden.

Der Product-Owner möchte, dass man die Listen mit einer Suchfunktion filtern kann und das Öffnen des Bearbeiten-Menüs sollte durch einen Doppelklick, nicht durch einen Button, erfolgen.

Retrospektive

Es gab im aktuellen Sprint keine Probleme im Prozess und deshalb auch keine Verbesserungsvorschläge.

Planning Meeting 1

Es sollen folgende User Stories implementiert werden:

- Zimmerbelegung anzeigen:
  Als Geschäftsführer will ich in die Zimmerbelegung einsehen.
- Suchfunktion in Anzeige:
  Als Rezeptionist will ich über eine Suche schnell zu gewünschten Kunden, Zimmern oder Reservierungen gelangen.
- Öffnen des Bearbeiten-Menüs nach Doppelklick:
  Als Rezeptionist will ich durch einen Doppelklick auf einen Datensatz, diesen bearbeiten.

Planning Meeting 2

Die User Stories werden folgendermaßen implementiert:

- Die Zimmerbelegung wird durch ein GUI mit durch eine Tabelle, in der die Zellen bei freien Tagen grün und bei belegten Tagen rot angezeigt wird, dargestellt.
- Kunden-, Zimmer- und Reservierungslisten GUI erhält eine Suchleiste, die bei einer Eingabe die Liste filtert
- Kunden-, Zimmer- und Reservierungslisten GUI wird ein Doppelklick-Event zum Bearbeiten der Einträge hinzugefügt

Bug-Fix von:

- Eingabe-Validierung für alle Editoren.

12.01.2011 - Sprint 6

Review und Retrospektive

Alle must-have Features wurden implementiert und erfolgreich getestet. Das Projekt wurde vom Product-Owner abgenommen, somit wurde das Projekt erfolgreich beendet.

18.01.2012 – Endpräsentation

Das fertig gestellte Produkt wurde präsentiert.

Retrospektive

Prozessbewertung

Das vorgestellte Projekt wurde von dem Team, bestehend aus vier Personen, mittels eines agilen Vorgehensmodells abgewickelt. Agilität wird durch eine flexible Vorgehensweise und Dynamik umschrieben und eignet sich bei wechselnden Anforderungen, wie es bei der Software Entwicklung der Fall ist, sehr gut. Außerdem sollte der Prozess sehr ergebnisorientiert ausgelegt werden, um der Definition der Agilität gerecht zu werden.

Selbst im Verlauf dieses relativ kleinen Projektes konnte das Team diese angeführten Eigenschaften erkennen und dadurch auf wechselnde Anforderungen recht schnell einstellen. Des Weiteren konnte das Team auch sehr viel eigene Initiative bei Veränderungen zeigen, wodurch eine nicht unerhebliche Menge an Verbesserungen in das schlussendliche Produkt einflossen. Nachteile konnten im Bezug auf die Eigenschaft der Agilität keine erkannt werden.

Die Punkte des agilen Manifestes, welche die Grundlage für jeden agilen Entwicklungsprozess bilden, konnte in den Phase des Projektverlaufs einmal mehr und an anderer Stelle etwas geringer erkannt werden. Vor allem der Grundsatz, dass der Mensch wichtiger ist als die einzelnen Prozesse kam vermehrt zur Anwendung.Beispielsweise mussten im Verlauf des Projektes einige Methoden, Technologien oder sogar Teile des Entwicklungsprozess von den Teammitgliedern geforderten Anpassungen unterzogen werden, was sich in der Folge als positiv auswirkte. Eine weiteres Merkmal unseres Entwicklungsprozesses sind die lauffähigen Systeme. Dieser tiefgreifende Punkt

beeinflusste den gesamten Projektverlauf grundlegend in Planung und Ausführung. Die Planung wurde durch die Anforderung eines funktionsfähigen Systems etwas erschwert. Besonders die ersten Phasen der Entwicklung wurde dadurch mit viel Arbeit belastet, wogegen am Ende des Projektes kaum noch Aufgaben zu erledigen waren. Die gewonnen Zeit konnte allerdings durch weiteres Fehler-korrigieren genutzt werden. Weitere Zeit konnte im Projekt auch durch das nächste Merkmal gewonnen werden: keine umfassenden Dokumentation. Das Team, welches ansonsten häufig in seinen akademischen Projekten unter der Dokumentationslast leidet, freute sich wiederholt, dass man sich auf die Programmierung konzentrieren konnte. Durch eine saubere Architektur und die gute Kommunikation innerhalb des Teams behielten die Mitglieder einen guten Überblick über das Projekt. "Reagieren ist wichtiger als den Plan befolgen" ist einer der wichtigsten Grundsätze bei agilen Vorgehensmodelle. Dabei werden die Spontanität und die Kreativität des Projektteams auf die Probe gestellt. Dies war auch bei unserem kleinen Projekt der Fall als Anforderungen geändert oder neue Funktionen gefordert wurden. Allerdings konnte dies vom Team und auch vom Prozess gut aufgenommen werden und das Projekt konnte erfolgreich zum Abschluss gebracht werden. Weitere Merkmale von agilen Entwicklungsmethoden wie "enger Kundenkontakt" oder "keine umfassenden Verträge" kamen bei unserem Projekt nicht zum tragen, da der Kunde durch die LVA-Leitung dargestellt wurde und daher ein reger Kontakt nicht immer möglich war.

Diese angeführten Merkmale werden vollständig durch den Entwicklungsprozess SCRUM, der auch in unserem Projekt angewandt wurde, erfüllt. Abgerundet werden die Merkmale von SCRUM durch die spezifischen Eigenschaften der "kurzen Iterationen" und "Team hat Freiheiten beim erreichen der Ziele", welche auch in unserem Projekt zum tragen kamen.

In den folgenden Abschnitten werden kurz einige Anpassungen des Prozesses erläutert, welche im Lauf des Projektes vorgenommen wurden.

Anpassung 1 – Verzicht auf Daily Scrum

Die erste Anpassung wurde bereits beim ersten Meeting des Teams von einigen Teammitgliedern vorgeschlagen, das Daily Scrum nicht abzuhalten, da der Nutzen in diesem kleinen Projekt als beschränkt eingeschätzt wurde und aufgrund der Größe von Projekt und Team eine indirekte Kommunikation bei Problemen als ausreichend erachtet wurde. Dazu gab es zu nächst einige Einwände, wie was bei Problemen oder Unklarheiten im Sprint gemacht werden soll. Doch nach einiger Diskussion konnten alle Befürchtungen ausgeräumt werden. So wurden Problemen im Verlauf eines Sprints in der ersten Phase mittels Email-Verkehr, später dann anhand der Kommentare von Issues in Github gelöst.

Da alle Teammitglieder sehr gewissenhaft diese Anfragen schnell und präzise beantworteten, kam es zu keinerlei Verzögerungen. Der Wegfall der Daily Scrums konnte gut kompensiert werden, wodurch keine neue terminliche Belastung der Teammitglieder bestand.

Anpassung 2 - Rollen Kunde, Benutzer und Management werden vernachlässigt

Eine weitere Anpassung, die Vernachlässigung von Management-, Benutzer- und Kunde-Rolle, wurde ebenfalls am Beginn des Projektes festgelegt. Alle drei Rolle gehören ohnehin nicht direkt zum Scrum-Team und werden trotzdem oft als Rollen mit eigenen Aufgaben in diesem Vorgehensmodell erwähnt. So ist das Management für die Bereitstellung der Rahmenbedingungen zuständig, was in unserem Fall teils durch die LVA-Leitung, teils durch das Team selbst veranlasst wurde und somit kein Problem darstellte.

Der Benutzer soll im eigentlichen Scrum-Prozess das Produkt aus seiner Sicht beurteilen. Für die Hotelsoftware, die in unserem Projekt erzeugt wurde, konnte allerdings kein Kunde angefragt werden. Die Definition dieser Rolle und deren Simulation hätte, unserer Meinung nach, sehr viel Zeit in Anspruch genommen und dabei keinen nennenswerten Nutzen mit sich gezogen.

Der Kunde stellt bei Scrum normalerweise eine sehr wichtige Rolle dar, da er in regen Kontakt mit dem Product Owner steht und so seine Anforderungen in das Projekt einfließen lassen kann. Den Kunde stellte in unserem Fall die LVA-Leitung dar, welche anfangs die Anforderungen an das System formulierte. Allerdings konnte kein reger Kontakt im Umfeld der Universität aufgebaut werden, wodurch der Product Owner einen Teil des Kunden simulierte und dadurch einige Aspekte dieser Rolle in das Projekt einarbeitete. Ein explizite Erwähnung oder Definition der Kunde-Rolle wurde allerdings nicht beachtet, wodurch auch keine nennenswerten Nachteile entstanden.

Anpassung 3 – Jeder ist Entwickler

Eine der ersten Entscheidungen im Projektverlauf war die Aufteilung der Rolle unter den Teammitgliedern. Da sich einige Teammitglieder möglichst wenig mit der Dokumentation befassen wollten und einer bereits etwas mehr Erfahrung mit Scrum hatte, waren die Rollen schnell verteilt. Bei der Verteilung der Aufgaben wurde schon etwas länger diskutiert, da ja der Grundsatz der Gleichverteilung von Arbeit gelten sollte. So kam es, dass beschlossen wurde dem Scrum Master und Product Owner zusätzlich als Entwickler einzubeziehen, da ihre Aufgaben dem Team doch etwas begrenzt schienen. Scrum Master und Product Owner haben somit im Projektverlauf zwar hauptsächlich die Aufgaben ihren Rollen entsprechend übernommen, wurde aber kontinuierlich in die Entwicklungsarbeiten einbezogen.

Daraus konnten im Verlauf des Projektes keine negativen Aspekte abgeleitet werden, sondern im Gegenteil ergaben sich dadurch einige Vorteile. So hatten das gesamte Team einen sehr genauen Überblick über den Projektstand und etwaige Probleme. Außerdem bedarf es bei den Meetings keinen ausschweifenden Erklärungen bei Problemen oder Änderungen, was dazu führte, dass diese recht zügig und unkompliziert von Statten gingen.

Anpassung 4 – Kürzere Sprintdauer als vorgeschlagen

Wir haben uns zu Beginn des Projekts dafür entschieden, dass die Sprintdauer lediglich eine Woche betragen soll. Ein Grund dafür war, dass wir durch den Verzicht auf das Daily die Gefahr sahen, dass wird bei langen Sprints den Projektfortschritt aus den Augen verlieren könnten. Wir befürchteten zudem, dass die Arbeit in den ersten Wochen unerledigt bleiben würde und erst in den letzten Sprinttagen damit begonnen wird. Durch die kurze Sprintdauer sollte – in Verbindung mit einem großen wöchentlichen Meeting (siehe folgender Abschnitt) – gewährleistet werden, dass die beschriebenen Probleme nicht auftreten. Im Laufe des Projekt erwies sich dieses Vorgehen als äußerst praktikabel. Die kurzen Sprints konnten gut überblickt werden und Abstimmung während des Sprints war kaum erforderlich (fast ausschließlich im Sprint Planning). Die Arbeit wurde außerdem regelmäßiger und kontinuierlicher durchgeführt (nicht zwei Wochen nichts und dann alles auf einmal).

Zudem war dies auch hinsichtlich des Ziels der Lehrveranstaltung förderlich, da ja nicht das Produkt an sich, sondern viel mehr der Prozess im Vordergrund steht. Durch die höhere Sprintanzahl konnte mehr Erfahrung über den Prozess gewonnen werden.

Anpassung 5 – Ein großes Meeting

Die Anpassung 4 entwickelte sich im Laufe des Projektes immer weiter, so wurde zu Beginn lediglich Entschieden, dass es zwei Meetings in der Woche geben sollte. Das Team erachtete diese Maßnahme auf Grund der Anpassung 3 - Jeder ist Entwickler als gerechtfertigt. Allerdings kam es bereits in der ersten Woche zu erheblichen Problemen, da es im Studentenleben nahezu unmöglich war zwei Treffen innerhalb einer Woche zu organisieren. Schlussendlich wurde daher die Entscheidung getroffen, dass nur einen Termin pro Woche eingeplant und bei diesem Treffen alle vier Scrum-Meetings – die da wären Sprint Planning 1, Sprint Planning 2, Sprint Review und Retrospektive – abzuhalten. Aufgrund einer guten Atmosphäre im Team, einer strikten Meeting-Durchführung und detaillierten Protokollierung kam es dabei zu keinerlei Problemen. Der Overhead des Projektes konnte damit erneut reduziert werden.

Anpassung 6 – Optimierung der Artefakte

Die Dokumentation in Scrum fällt deutlich geringer aus als bei anderen Entwicklungsprozessen, trotzdem können auch hier einige Hindernisse entstehen. Am Beginn des Projektes hat sich jedes Mitglied um seine Aufgabe gekümmert, dabei kam es dazu, dass für die Erstellung der Artefakte unterschiedlichste Technologien gewählt wurden. Außerdem wurden alle Artefakte, welche in Scrum gefordert wurden berücksichtigt. Einzige Ausnahme stellte die Produktvision dar, wobei dies im Charakter des Projekts begründet liegt (Produkt war vorgegeben). Product Backlog wurde in Latex verfasst, wogegen Sprint Backlog, Burndown-Charts, Impediment Backlog und Definition of Done in Excel erarbeitet wurden.

Das Aktualisieren der Excel-Artefakte stellte sich schnell als äußerst zeitintensiv heraus, wobei wir dafür weniger einen Mangel an Disziplin als die gewählte Technologie verantwortlich machten. Als nächsten Schritt wurden daher die Artefakte durch eine andere Technologie erfasst (siehe Kapitel Prozessbeschreibung Abschnitt Technologin). Github bot dazu die nötigen Funktionen, so wurde der Product Backlog im Wiki von Github eingebettet und der Sprint Backlog und Definition of Done konnte durch die Issues realisiert werden.

Als Resultat dieser Änderung stieg das Commitment hinsichtlich der laufenden Aktualisierung der Daten spürbar an. Als weiterer Bonus wurde dadurch auch die Kommunikation vereinfacht, da diese nahezu komplett von Github übernommen werden konnte.

Fazit

Scrum hat sich als Entwicklungsprozess hervorragend für unser kleines Projekt geeignet. Die mitgebrachten Eigenschaften haben sich großteils in der Praxis bewährt und durchaus einen positiven Effekt auf das Endprodukt ausgeübt. Durch kleinere Änderungen konnte der Prozess präzise an die vorgegeben Anforderungen und somit an das Projekt angepasst werden, wodurch die Effizienz des Vorgehensmodells deutlich erhöht werden konnte.

In den folgenden Abschnitte zeihen die einzelnen Teammitglieder ein persönliches Fazit über den Projektverlauf mit besonderem Augenmerk auf den Entwicklungsprozess und dessen Anpassung.

Teilnehmer 1 – Alexander als Entwickler

Als Entwickler konnte ich Scrum sehr viel positives abgewinnen, da ich in meinen Aufgaben einen hohen Grad an Freiheit empfand. Außerdem konnten wir durch die unterschiedlichsten Anpassung die einzelnen Arbeiten etwas erleichtern.

Als negativ empfand ich die vielen Meetings am Anfang des Projektes, welche wir allerdings schnell reduzieren konnten.

Teilnehmer 2 – Thomas als Entwickler

Der Prozess selber hat sich anfangs nur bedingt für unser konkretes Projekt geeignet, da sich im universitären Umfeld abspielte. Allerdings kann ich mir sehr gut vorstellen, dass es sich im Projektalltag der einzelnen Unternehmen deutlich besser eignen würde. Vor allem die Agilität und das aufkommende Teamgefühl könnten sich sehr positiv auf einige Projekte auswirken.

Im konkreten Fall konnten wir durch einige Anpassungen des Entwicklungsprozessen das Projekt schlussendlich zum Erfolg bringen.

Teilnehmer 3 – Stefan als Scrum Master

Anfangs habe ich den anfallenden Koordinationsaufwand mit Meetings und Artefakten etwas unterschätzt, da Scrum als leichtgewichtiger Entwicklungsprozess gilt. Dies konnten wir durch einige Anpassung recht zügig

in den Griff bekommen. Schlussendlich können wir mit dem Projektverlauf recht zufrieden sein, besonders das Gefühl im Team etwas zu erreichen zog sich durch das gesamte Projekt, was ich unter anderem als positive Erfahrung aus dem Projekt mitnehme.

Teilnehmer 4 – Martin als Product Owner

Die Rolle des Projekt Owners war neu für mich, deshalb fand ich es etwas schade, dass ich diese nicht vollständig ausleben konnte, da der Kontakt mit dem Kunde nicht vorhanden war. Dies stellt allerdings die Hauptaufgabe der Rolle dar. Ansonsten konnten wir die Vorteile des Entwicklungsprozesses Scrum detailliert kennen lernen. Besonders die Anpassungsfähigkeit von Scrum hat mich überrascht und hat sich des weiteren äußerst positiv auf das Projekt ausgewirkt.

Softwareprodukt

In diesem Kapitel wird das von uns erstellte Software-Produkt beschrieben, wobei vorallem auf Use Cases, die Architektur und die verwendeten Technologien eingegangen wird.

Einleitung

Beschreiben Sie überblicksmäßig die Rahmenbedingungen der Implementierung sowie die Implementierung selbst.

Unsere Implementierung der Hotel-Reservierungs-Lösung wurde vom Projektteam unter dem Arbeitstitel „BlueHotel“ geführt. BlueHotel ist für kleine Hotels (Familienbetriebe, Urlaub am Bauernhof, Ferienhäuser) konzipiert, in bei denen es eine überschaubare Anzahl an Räumen, und keine Raumklassen gibt (d.h. statt zum Beispiel 200 freie Räume in der Klasse „Premium“ und 100 freie Räume in der Klasse „Business“ gibt es bei uns im Hotel dedizierte Räume, die eigenständige Namen haben, und nur einzeln existieren, zum Beispiel „Präsidenten-Suite“, „Garçonnière A“ und „Das Loft“). Aus diesem Grund wird auch jeder Raum eigenständig in der Datenbank erfasst.

Als Rahmenbedingung gilt zusätzlich, dass in einem Hotelzimmer jeweils 1-3 Personen untergebracht sind, wobei sich dieses beliebig aus Erwachsenen und Kindern zusammensetzen können (mit der Ausnahme, dass kein Kind alleine ein Zimmer belegen kann). Für alle Kombinationen (bis zu 6: 1E, 2E, 3E, 1E+1K, 1E+2K und 2E+1K) von Erwachsenen und Kindern kann jedem Zimmer ein eigener Nächtigungspreis zugewiesen werden.

Weiters gehen wir davon aus, dass diese Lösung auf einem Einzelplatzrechner verwendet wird, und es keine Anforderung gibt, die Daten über das Web zugänglich zu machen.

Als zusätzliche Einschränkung gehen wir davon aus, dass die Preise keiner saisonalen Schwankung unterliegen – das User Interface wurde trotzdem so konzipiert, dass man die Preise der einzelnen Zimmer leicht manuell ändern kann, eine einmalige Eingabe der Preise für jede Saison mit automatischer, zeitlicher Anpassung wurde nicht vorgenommen.

Als UI-Sprache haben wir Englisch gewählt, durch ein Folgeprojekt könnte Lokalisierung (l10n) bzw. Internationalisierung (i18n) durch Java-eigene Technologien leicht hinzugefügt werden.

Use Cases

Im Folgenden wird die Implementierung beispielhaft anhand von ausgewählten, repräsentativen Use Cases inklusive Screenshots erklärt.

Kunde anlegen

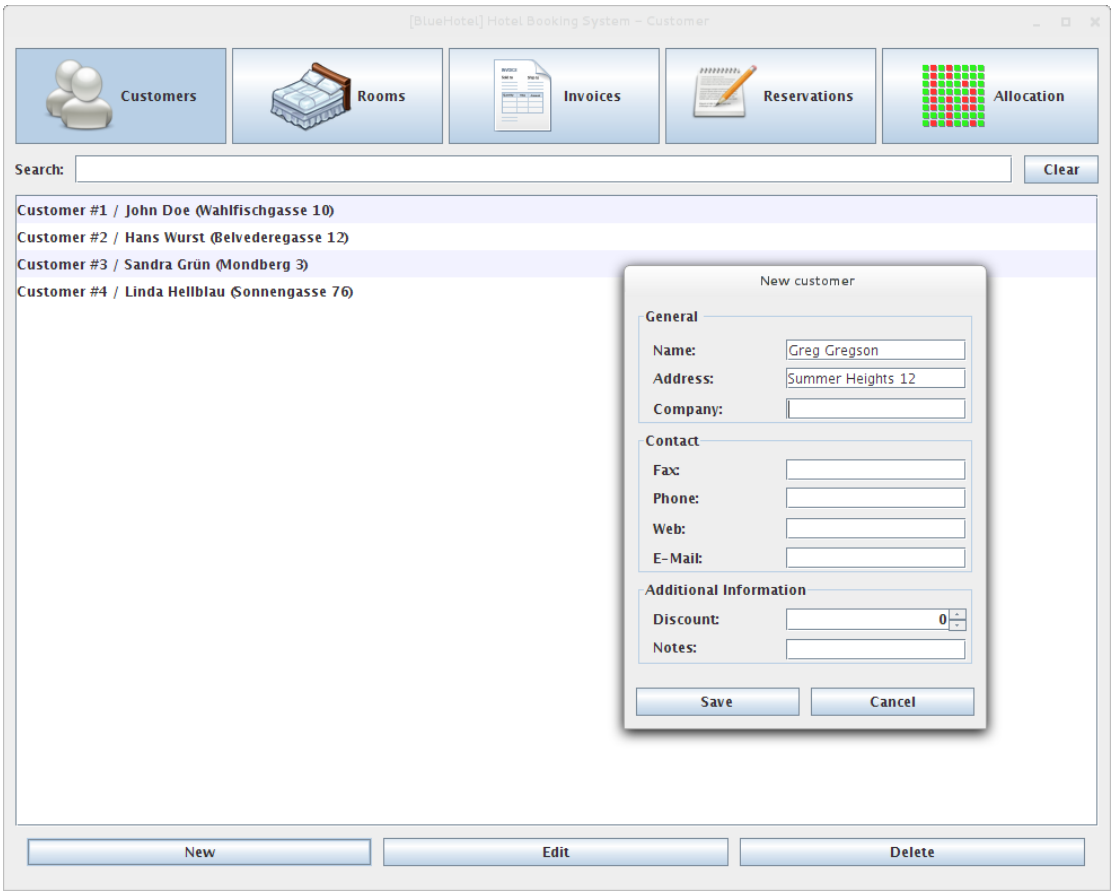


Abbildung: Kunden-Liste mit „Kunde anlegen“-Dialog

Das Anlegen von Kunden erfolgt über den Menüpunkt „Customers“. Diese Ansicht hat die selbe Struktur wie andere Listen im Programm:

- Such- und Filterleiste oben, mit „Clear“-Button
- Liste der Kunden, sortiert nach Erstellungsdatum
- C(R)UD-Buttons unten: New (=Create), Edit (=Read and Update) und Delete (=Delete)

Name und Adresse sind Pflichtfelder, die anderen Felder sind optional. Dies erlaubt ein schnelles Erfassen ankommender Kunden, die Kontaktdaten können dann optional nach der Ankunft eingetragen werden. Bei fehlerhaften Eingaben wird eine Fehlermeldung angezeigt, wenn der „Save“-Button aktiviert wird.

Raum anlegen

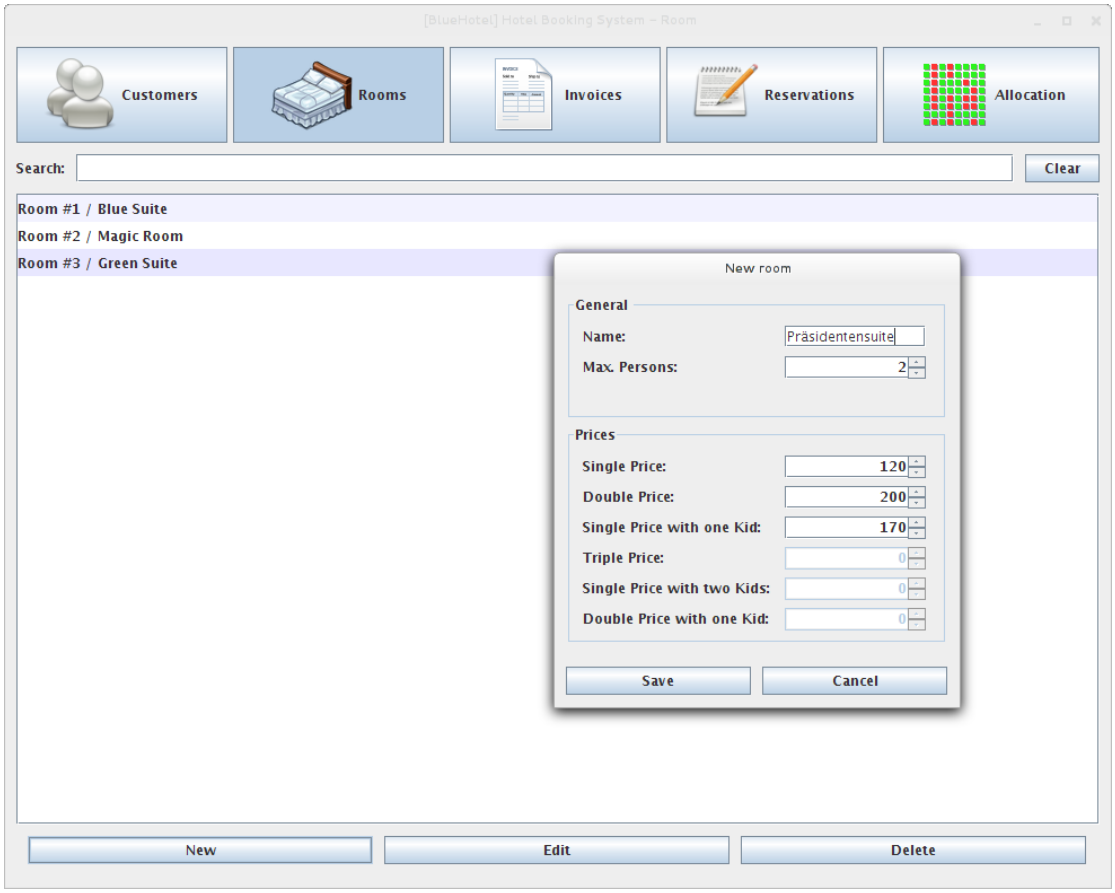


Abbildung: Zimmer-Liste mit „Raum anlegen“-Dialog

Um Zimmer anzulegen, bzw. die Preise zu editieren kann der Menüpunkt „Rooms“ verwendet werden. Hier haben wir als Pflichtfelder zum einen den Namen des Zimmers und die maximale Anzahl der Personen, die in diesem Zimmer Platz finden.

Je nachdem, welchen Wert das Feld „Max. Persons“ hat, werden im „Price“-Abschnitt des Dialogs unterschiedliche Eingabefelder freigeschalten. Alle freigeschaltenen Pflichtfelder müssen mit einem positiven Preis ausgefüllt werden, ansonsten erscheint einer Fehlermeldung.

Beispiel: Bei Auswahl von maximal 2 Personen müssen die Felder „Single Price“, „Double Price“ und „Single Price with one child“ ausgefüllt werden- die restlichen Felder (die nur bei 3 Personen relevant sind) sind deaktiviert, und können nicht ausgefüllt werden.

Auch hier erscheint wieder eine Fehlermeldung beim Speichern, wenn eine Input-Validierung fehlschlägt. Die Fehlermeldung gibt klar Auskunft darüber, welche Felder den Fehler verursacht haben, und wie der Fehler zu beheben ist.

Reservierung erstellen

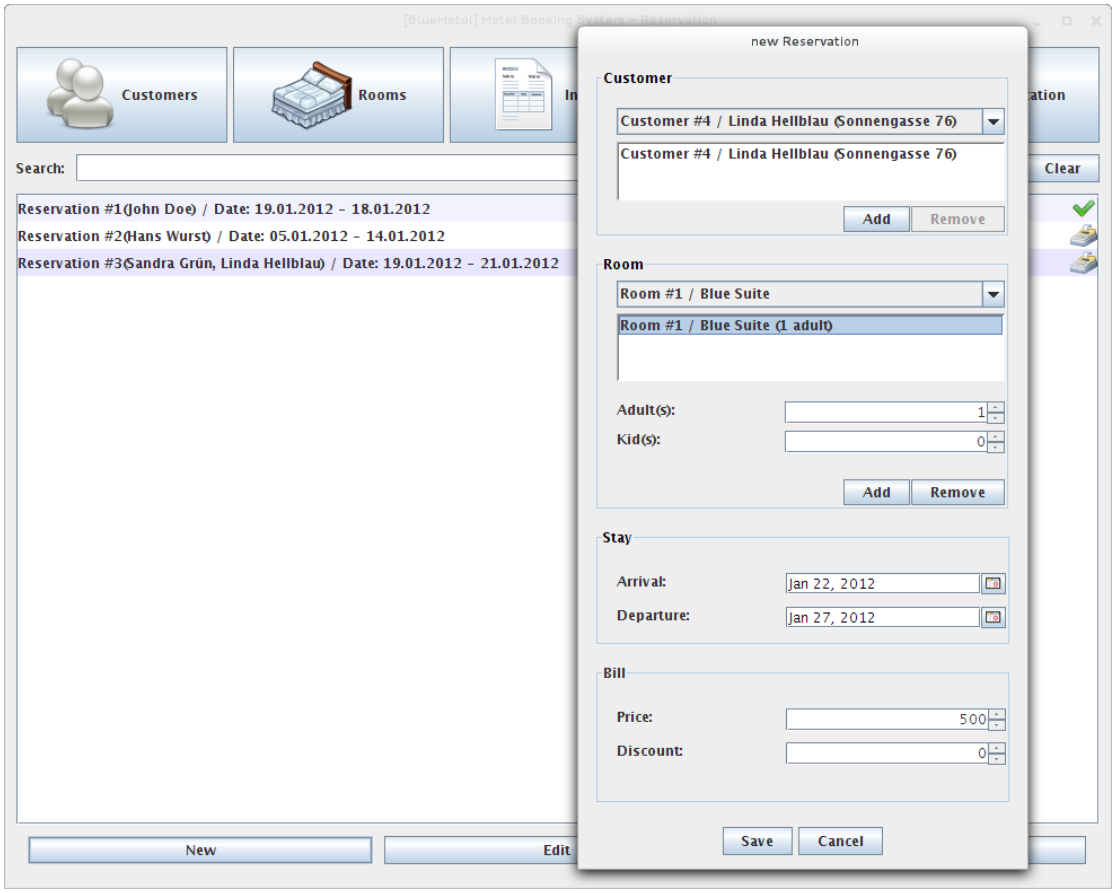


Abbildung: Reservierungs-Liste mit „Reservierung anlegen“-Dialog

Die am meisten benutzte Funktion betrifft das Verwalten der Reservierungen. Um einen möglichst reibungslosen Workflow zu gewährleisten, ist diese Ansicht auch die Start-Ansicht, wenn man das Programm öffnet.

Zum Erstellen einer Reservierung wählt man zuerst die Kunden aus, die in dieser Reservierung inkludiert sein sollen. Danach kann man die Zimmer verbuchen, und dabei die Anzahl der Erwachsenen und Kinder angeben. Falls man einen Raum überbucht, wird das als Fehlermeldung beim hinzufügen des Raums zur Reservierung angezeigt. Nach der Zuweisung der Zimmer kann nun per Date-Picker das Start- und Ende-Datum des Aufenthalts angegeben werden.

Basierend auf der Raum-Auswahl wird dann der Preis berechnet. Dieser Preis kann bei Bedarf vom Ersteller der Buchung noch verändert werden – ein auf der Rechnung ausgewiesener Rabatt (ohne Minderung des Originalpreises) kann hier ebenfalls eingegeben werden.

Fehler in der Eingabemaske werden wie schon zuvor beim Speichern entdeckt und per Fehlermeldung angezeigt.

Rechnung erstellen

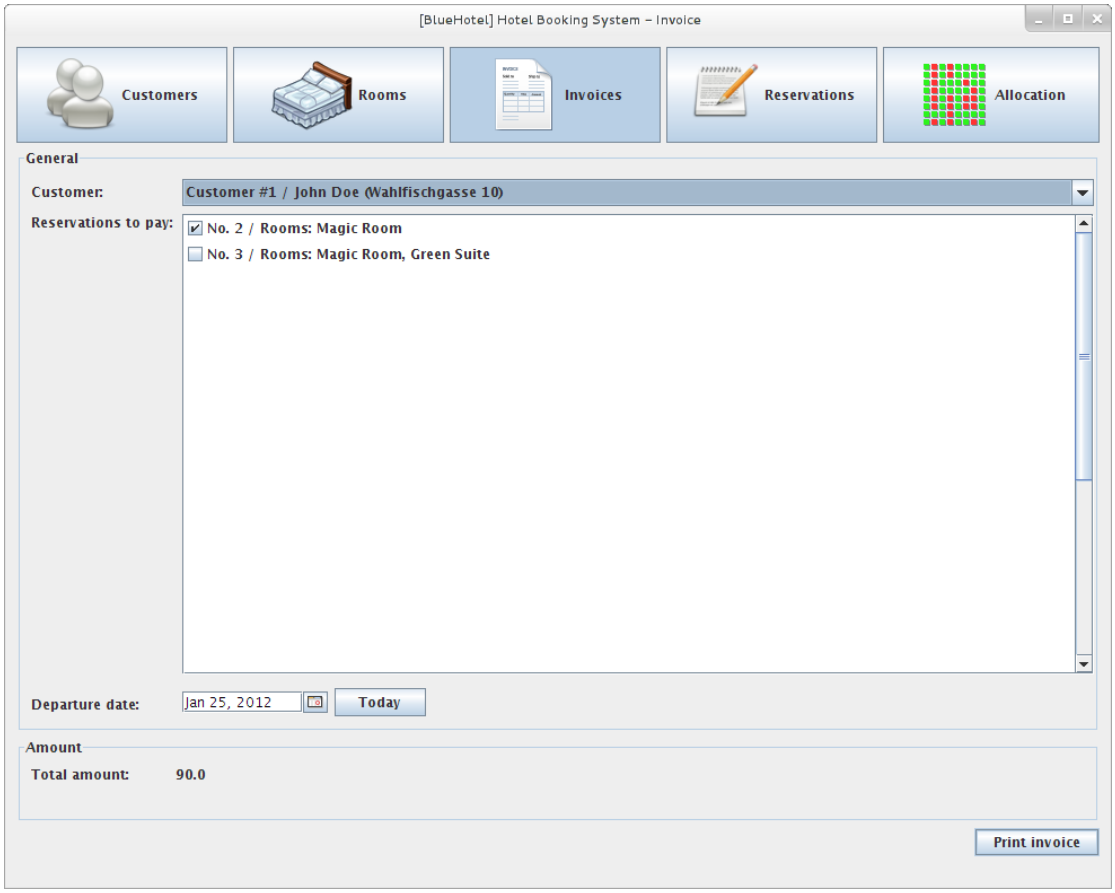


Abbildung: Rechnung erstellen mit Kunden- und Reservierungs-Auswahl

Die Rechnungslegung ist ein wichtiger Bestandteil des Systems, denn das bisherige manuelle Erstellen von Rechnung ist mühsam, und war einer der Gründe für die Beauftragung eines neuen Software-basierten Systems zum Management der Zimmer, Reservierungen und Rechnungen.

Ausgangspunkt für unsere Designentscheidungen sind wieder die üblichen Szenarien von kleinen Hotels: Ein Kunde nähert sich der Rezeption und will eine oder mehrere Reservierungen bezahlen – dies können auch Reservierungen sein, die nicht vom Kunden selbst gebucht worden sind. Als Beispiel sei dem Leser hier ein Firmenausflug vor Augen geführt, bei dem die Angestellten selbstständig ihre Zimmer reservieren, die Endabrechnung dann aber direkt vom Buchhalter übernommen und gesammelt bezahlt wird.

Weiters ist es möglich, das Abreisedatum – und somit das Rechnungsdatum – festzulegen. Für den unwahrscheinlichen Fall dass die Gäste verfrüht abreisen, steht ein „Today"-Button zur Verfügung. Die Reservierungen werden beim Erstellen der Rechnung wenn nötig automatisch verkürzt.

Rechnungen werden im HTML-Format gespeichert und angezeigt.

Rechnung einsehen

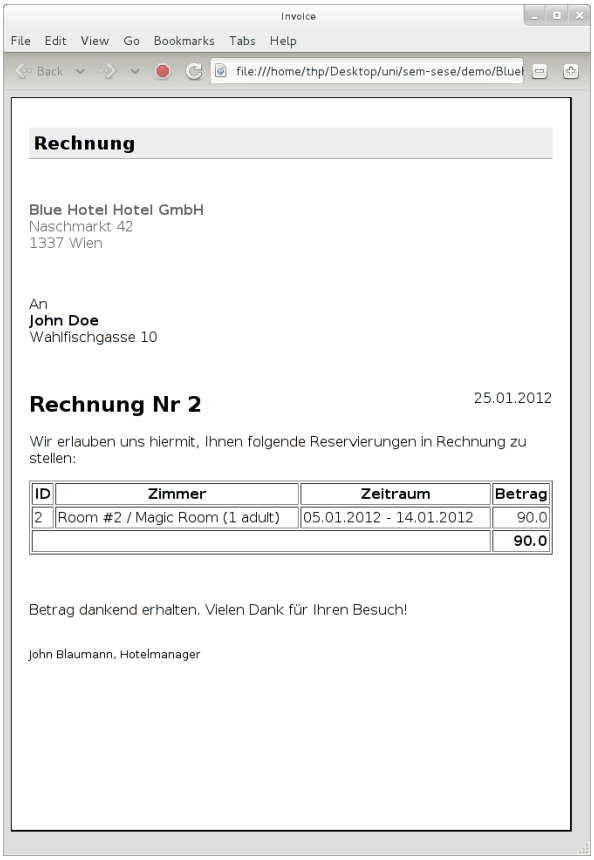


Abbildung: Erstellte Rechnung, direkt im Dateisystem abgelegt und geöffnet

Wie im vorherigen Abschnitt bereits erwähnt ist die Rechnungslegung ein vitales Instrument zur Aufrechterhaltung des Hotelbetriebs aufgrund Geldeinnahmen durch Kundenzahlungen.

Nachdem man im Menüpunkt „Invoices“ (Rechnung erstellen) die Funktion „Print Invoice“ (Rechnung drucken) aktiviert hat, wird automatisch eine Rechnung in der Datenbank angelegt, die betroffenen Reservierungen als „bezahlt“ markiert und eine ausdruckbare Form des Rechnung als HTML-Datei im Dateisystem abgelegt. Dies hat einige Vorteile – unter anderem können die Rechnungen so leicht archiviert werden, und sind auch ohne die Benutzung des Programms verfügbar – was vorallem bei einer Vorhaltedauer von Rechnungen von 7 Jahren durchaus hilfreich ist.

Die Rechnungs-Informationen werden aus den Kunden- und Reservierungsdaten generiert, wobei hier auch Kundenrabatte (zB bei Stammgästen) Berücksichtigung finden.

Um den Ausdruck der Rechnung weiter zu beschleunigen, wird nach dem Speichern des HTML-Dokuments dieses sofort im Standard-Webbrowser des Systems geöffnet.

Raumbelegung anzeigen

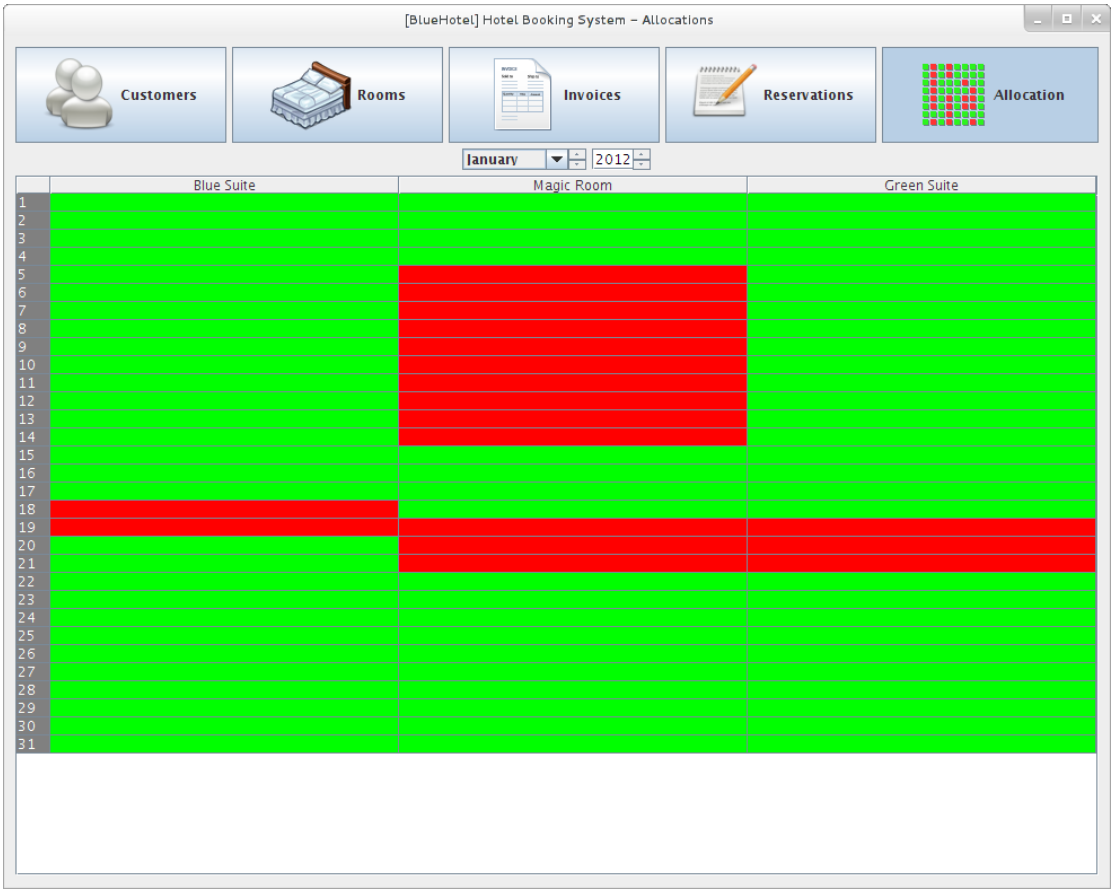


Abbildung: Anzeige der Raumbelegung für den Monat Jänner 2012

Für einen guten, visuellen Überblick über die momentane Auslastung der einzelnen Zimmer haben wir zusätzlich eine Belegungs-Liste der Zimmer implementiert. Diese zeigt optisch ansprechend die Reservierungen pro Zimmer und Tag an. So lässt sich auf einen Blick erkennen, wann noch Zimmer frei sind – dies ist vorallem bei Telefon-Anfragen („Wann haben Sie im März noch ein Zimmer frei?“) hilfreich.

In der aktuellen Implementierung dient die Belegungs-Grafik rein zur Visualisierung der Belegungen. Für zukünftige Erweiterungen wurde das System so flexibel gehalten, dass es möglich sein wird, die Reservierungen per Doppelklick auf rote Bereiche zu öffnen.

Weiters wurde von uns ein nice-to-have Feature angedacht, bei dem man durch drag'n'drop im grünen Bereich der Ansicht eine neue Reservierung erstellen kann, bei der das Ankunfts- und Abfahrtsdatum (Zeilen) sowie das gewählte Zimmer (Spalte) bereits vorausgefüllt sind. Dies wurde aber in der vorliegenden Version noch nicht implementiert.

Architektur

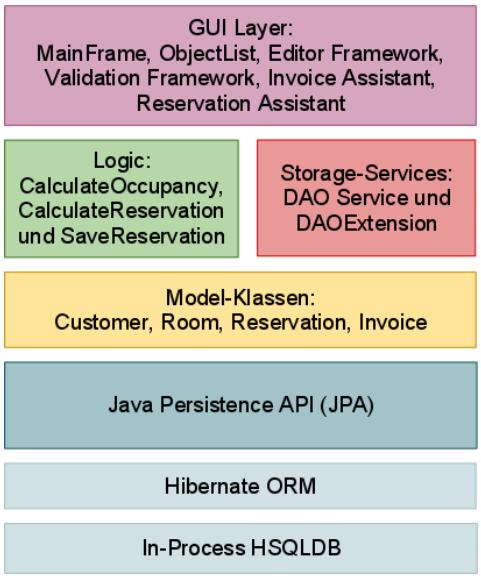


Abbildung: Architektur-Übersicht des Projekts BlueHotel

Beim Design der Software haben wir besonderen Wert auf eine gelayerte Architektur gelegt – im Idealfall greift ein Layer nur auf die Services des eigenen Layers und des direkt darunter liegenden Layers zu. In unserem Projekt verwenden wir als Datenbank eine In-Process HSQLDB, die gleichzeitig mit dem Programm gestartet wird.

Als ORM-Layer kommt Hibernate zum Einsatz; das Ansprechen des ORM-Layers erfolgt durch die standardisierte Java Persistence API (JPA). Die Model-Klassen verwenden die JPA mittels Annotationen. Die Model-Klassen werden von den Storage-Services (DAO mittels Generics plus DAOExtension für spezialisierte Abfragen) in die Datenbank geschrieben und von dieser gelesen.

Weiters wurde vom Projektteam ein spezielles Logic-Modul entwickelt, das parallel zu den DAO-Services Funktionalitäten überhalb der Datenbank zur Verfügung stellen, die aber nicht von der GUI abhängig sind. So eine Lösung hat den Vorteil, dass in einem möglichen Nachfolgeprojekt, bzw. einer möglichen Erweiterung der UI-Layer ausgetauscht werden kann, ohne die Applikations-Logik neu zu schreiben.

Im Architekturdiagramm zur bessern Lesbarkeit nicht abgebildet sind einige Hilfsklassen und Relations-Tabellen, die für die korrekte Persistierung von Objekten und als Glue-Code zwischen den Modulen und Schichten eingesetzt werden.

Technologien

Nach ursprünglichen Überlegungen und Besprechungen im Team (beim ersten Meeting) haben wir – unter Berücksichtigung des technischen Backgrounds aller Teammitgleider – uns für Java-Technologie als kleinsten gemeinsamen Nenner entschieden.

Weiters haben wir uns mit HSQLDB und Swing auf uns bekannte Technologien zur Datenspeicherung, bzw. zum UI-Design entschieden, da das Einlernen in andere Technologien den Projektfortschritt nur unnötig verlangsamt.

Als Testing-Framework haben wir uns bei Unit-Tests auf JUnit4 geeinigt, die funktionalen Tests werden ausschließlich manuell durchgeführt.

Die Vorteile dieser Technologien sind ausschließlich:

- Erfahrung aller Projektmitglieder mit der Sprache Java
- Distributions-Möglichkeit als „self-contained“ JAR-Datei
- Tooling-Support mit Eclipse und WindowBuilder Pro
- Portabilität von Java SE (Windows, Mac OS X und Linux)

Die Nachteile der gewählten Technologien sind unter anderem:

- Wenig Flexibilität der Sprache Java im Vergleich zu dynamischen Programmiersprachen
- Komplizierte Formulierung des Datenbank-Modells mittels Annotations und XML-Dokumenten
- UI-Integration (optisch sowie im Verhalten) von Swing-UIs in die nativen Oberflächen von Windows, Mac OS X und Linux

Anhang

Tagebuch – Alexander Duml

04.11.2011

Ich war beim ersten Meeting abwesend, habe aber per Mail bescheid gekriegt was besprochen wurde. Ich bin zusammen Thomas im Entwickler-Team.

10.11.2011

Ich hab mir drei User Stories überlegt, diese im Latex-File angehängt und gepusht.

11.11.2011

Thomas und ich haben vor dem zweiten Meeting das Projekt eingerichtet. Wir haben uns HSQLDB heruntergeladen, da diese Datenbank leicht für das Projekt ausreicht. Wir verwenden Eclipse als Entwicklungsumgebung. Anschließend wurde das Repo eingerichtet. Im Programm sprechen wir die Datenbank über das Hibernate Framework an. Das Einrichten von Hibernate hatte einige kleine Komplikationen, da ich persönlich noch nie mit dem Framework gearbeitet hatte und es bei Thomas auch schon einige Zeit her war. Aber schlussendlich hat alles hingehauen.

11.11.2011

Beim zweiten Meeting haben wir unsere bisherigen Errungenschaften besprochen. Außerdem wurden neue Tasks vergeben, ich muss den CRUD Mechanismus + GUI für die Reservierungen implementieren. Für die Implementierung der GUI wurde von Stefan der Windows Builder Pro für Eclipse empfohlen.

17.11.2011

Meine Aufgabe ist es ein Model für Reservierungen anzulegen und anschließend das GUI, welches das Model aufruft zu implementieren. Die Programmierung des Models verlief problemlos, Hibernate ist ziemlich angenehm zu benutzen. Da wir beim Meeting letzte Woche besprochen haben, dass es angenehm wäre das GUI auch mit einem grafischen Tool zu designen und nicht alles runter zu coden. Deshalb hab ich den Windows Builder Pro (den ich vorher über ein Eclpse Software update geladen hatte) getestet. Ich muss sagen, dass ich mit dem Tool überhaupt nicht zurecht gekommen bin, es hat alles komplizierter gemacht als es anfangs war. Es wurde ständig alles verschoben, sodass ich sobald ich dieses GUI fertig gestellt hatte, das Plugin nicht mehr verwendet habe.

18.11.2011

Leider ist sich das Meeting zeitlich nicht ausgegangen, aber ich habe mich entschuldigen lassen und schon per E-Mail meine Fortschritte und Tätigkeiten mitgeteilt. Nach dem abgehaltenen Meeting habe ich gleich die Zusammenfassung des Meetings zugesandt bekommen. Ich soll als nächstes die Logik auslagern und die Reservierung mehrerer Zimmer ermöglichen.

Außerdem wird bis zum nächsten Mal die GUI refactored.

23.11.2011

Ich hab diese Woche die Zimmerreservierung verbessert, da es zurzeit nur möglich war für einen der mehreren Kunden ein Zimmer zu reservieren. Um das Implementieren zu können musste ich mich in einige Hibernate Dokumente einlesen, da ich dachte, dass es wahrscheinlich bereits eine Funktion für diese ManyToMany Beziehenung mit zusätzlichen Feldern bereits existiert. Damit das gewünschte Feature funktionierte brauchte es schlussendlich noch zwei extra Klassen. Es ergab sich dann noch ein weiteres Problem, dass die Speicher-Funktion in ObjectList nicht mehr ausreichte. Damit der Abspeichervorgang funktioniert muss zuerst ein Reservierung (Reservation) in die Datenbank gespeichert werden, damit man dem ManyToMany Objekt (RoomReservation) die ID der Reservierung mitgeben. Nach dem die RoomReservation auch gespeichert wurde muss das Reservation Objekt nochmals upgedatet werden. Außerdem wurde noch die Kostenberechnung der Reservierung in ein neues Paket (logic) ausgelagert.

25.11.2011

Das Meeting ist gut verlaufen, Sprint 2 erfolgreich abgeschlossen. Kleinere Bugs wurden in Sprint 3 übertragen.

29.11.2011

Ich hab es leider nicht geschafft beim Prozess Workshop anwesend zu sein.

02.12.2011

Beim fünften Meeting wurde Sprint 3 erfolgreich abgeschlossen. Da ich die Woche ziemlich beschäftigt war hatte ich nicht viel zu berichten.

Es gab den Vorschlag die ganze Dokumentation, also Bug Report, TODOs usw. auch auf GitHub zu verwalten.

Die Verwaltung von Bugs auf GitHub ist eine gute Idee, man kann sehr übersichtlich sehen, was getan wurde oder noch aussteht. Man kann Tasks kommentieren und diese per Commit Message schließen.

03.12.2011

Mit der Reservierung gab es in der Datenbank einige Probleme, da Hibernate nicht so funktionierte wie gedacht. Ich habe deshalb die ManyToMany Beziehung wieder überarbeitet. Zusätzlich hat die Reservierung noch ein Storno-Flag erhalten, welches für weitere Funktionalität gebraucht wird.

06.12.2011

Die erwünschte Validierung wurde der Rechnungs-GUI hinzugefügt, das ging sehr schnell u problemlos.

Die Speicherlogik von Reservierung wurde überarbeitet und in das Logik Modul verschoben.

09.12.2011

Im sechsten Meeting wurde der aktuelle Sprint abgeschlossen und neue Tasks verteilt. Zusätzlich zu den Programmierarbeiten soll auch noch die Präsentation für nächste Woche vorbereitet werden.

11.12.2011

Bugfix: Verhindern von mehrfach Reservierungen von Zimmer am selben Datum. In der vorherigen Version hatte sich ein Denkfehler eingeschlichen der jetzt behoben wurde. Außerdem wurde die Beachtung des Storno-Flags eingebunden.

12.12.2011

Kleine Änderungen an der Präsentationsvorlage unterzogen und Hinzufügen eines Screenshots.

Hoffentlich müssen wir nicht präsentieren. ;)

14.12.2011

Bei der heutigen Zwischenpräsentation mussten wir nicht Vortagen, wahrscheinlich dann aber bei der Endpräsentation...

16.12.2011

Beim siebten Meeting wurde der aktuelle Sprint abgeschlossen und die letzten Tasks zur Komplettierung der Feature-Liste verteilt.

18.12.2011

Mein Task ist es diesmal die Zimmerbelegung grafisch anzeigen zu lassen, deshalb hab ich ein bisschen im Internet recherchiert wie andere das so machen.

19.12.2011

Ich war mit den Ergebnissen der Recherche nicht ganz zufrieden, deshalb hab ich selber versucht irgendwas mithilfe von JTables und dem Java Calaender Plugin zusammenzubasteln. Ich dachte mir das wird zack-zack gehen, aber da hatte ich mich getäuscht.

20.12.2011

Ich bin froh, dass ich heute den Task der grafischen Darstellung der Zimmerbelegungen gelöst habe. Ich finde das Ergebnis schaut akzeptabel aus. Ich hoffe ich muss nie wieder JTables verwenden.

12.01.2012

Beim achten und letzten Meeting gab es nur noch wenig zu erledigen.

13.01.2011

Die HSQLDB Datenbank startet jetzt mit dem Aufruf von BlueHotel. BlueHotel wurde in eine runnable-jar exportiert und kann jetzt per Doppel-Klick aufgerufen werden.

16.01.2012

Bug-Fix: Hibernate hatte zwei identische Tabellen durch die Beziehung von Zimmer und Reservierung erstellt, durch das Einfügen einer kleinen Codezeile war das Problem bereits Schnee von gestern...

18.01.2012

Wir mussten heute bei der Endpräsentation vortragen, bzw. meine Kollegen Martin und Thomas haben das übernommen, sehr gut.

Tagebuch – Stefan Müller

04.11.2011

Heute erstes Meeting. Team wirkt motiviert und kompetent – ein Wunder. Es konnten bereits einige Dinge geklärt werden (Rollen, Technologien, teilw. Aufgaben,...). Ich darf den Scrum Master spielen, womit ich mich gut abfinden kann. Todo für alle: Die Mitglieder sollen sich mit Scrum und ihrer Rolle vertraut machen. Ich werde das nächste Meeting organisieren und mich über den Report schlau machen.

Wir haben uns entschieden kein Daily Scrum durchzuführen. Ist m.M. nur logisch, da dass einfach nicht zu realisieren wäre (alle haben anderen Stundenplan). Ist nur unnötiger Over Head. Wenn wir die Sprintdauer kurz halten und Probleme per Email kommunizieren sollte das auch gehen.

06.11.2011

Habe heute die alten Unterlagen von meinem CSM Kurs rausgekramt und ein wenig darin geschmökert. Alles paletti.

09.11.2011

Ich habe mich über die Inhalte des Reports schlau gemacht. Anscheinend soll es dazu eine Vorlage geben. Diesbezüglich habe ich eine Email an die LVA-Leitung gesendet. Keine Antwort. Typisch TU. Martin hat bereits ein GitHub Repository eingerichtet. Kenne mich damit zwar nicht aus, sieht aber ganz gut aus.

11.11.2011

Heute war unser zweites Meeting. Zuerst: Review und Retrospektive. Alle Arbeiten wurden erledigt. Probleme gabs keine. Jetzt können wir eigentlich mit dem Projekt „richtig“ starten. Sprint Planning: Team hat sich dazu entschieden grundlegende Funktionen im ersten Sprint umzusetzen (Kunde, Zummer, Reservierung anlegen). Ich werde mich als SM zuerst um das Tooling kümmern (Sprintbacklog [derzeit auf Papier] und Burndown [derzeit nicht vorhanden]). Ansonsten scheinen alle Scrum zu verstehen; keine Probleme diesbezüglich.

Wir haben im Sinne einer fairen Arbeitsteilung beschlossen, dass SM und PO auch entwickeln. Dies ist zwar nicht ganz SCRUM-like. In einer Realen Projekt wäre das auch Blödsinn (Unvereinbarkeit und Konfliktgefahr) aber in unserem Umfeld ist das m.M. OK.

13.11.2011

Habe das Sprintbacklog für den ersten Sprint erstellt. Wir haben uns im letzten Meeting für ein Excel-basierter Tooling entschieden. Derzeit enthält das Dokument hauptsächlich Dummy-Daten. Zukünftig: Wir sollten es beim Sprintplanning am besten gemeinsam befüllen (estimations/wann will man was machen/...). Hoffe, dass die Leute das dann auch laufend aktuell erhalten werden.

16.11.2011

Heute einen Blick ins Repository geworfen. Es hat sich was getan. Die Kollegen sind fleißig.

18.11.2011

Heute war das dritte Meeting: Sprint 1 erfolgreich abgeschlossen. Sprintplanning für zweiten Sprint: GUI Refactoring und Reservierung mehrerer Zimmer. Mehr haben wir nicht hineingenommen, da Martin diese Woche keine Zeit hat. Ist aber Ok, da wir gut in der Zeit liegen.

Problem: Wir haben nicht getestet. Habe dem Team erklärt, dass wir nach jedem Sprint ein potentiell auslieferbares Stück software haben sollten. Es muss also auch schon getestet sein! Wir haben uns darauf geeingt das Testen nun mit einzuplanen. Meine Aufgabe: Templates für Testing erstellen.

19.11.2011

Heute erstellt: Template für Bug-Tracking und Testfälle und Testfälle für CRUD Kunde. Außerdem: Tests für CRUD Kunde durchgeführt. Scheint insgesamt gut zu funktionieren; lediglich zwei Fehler gefunden und eingestellt.

21.11.2011

Problem: Die Kollegen tragen ihren aktuellen Arbeitsstand nicht regelmäßig in das Sprint Backlog-Excel ein. Werde das Ansprechen...

23.11.2011

Liebes Tagebuch! Heute habe ich ein refactoring der GUI durchgeführt. Menü und ObjectList sind nun in einem Frame vereint. Editieren ist weiterhin ein eigener Dialog. Scheint so aber zu passen.

25.11.2011

Heute war das vierte Meeting: Abschluss Sprint 2. Teilweise noch Dinge aus Sprint 2 offen (v.a. Bugs) und in Sprint 3 übernommen. Hoffe, dass das nächsten Sprint besser läuft. Besonder die Bugs hätten eigentlich gefixed gehört. Habe das entsprechend deutlich gemacht. Anschließend: Sprint 3 geplant.

26.11.2011

Diskussion über das Tooling: Exceln ubzudaten wird als umständlich empfunden (gebe den Kollegen recht!). Haben uns eine andere Lösung überlegt: Ab nun Verwenden wir die Issues aus GitHub um die Aufgaben zuzuweisen. Damit können wir das Sprint Backlog abbilden!

28.11.2011

Habe heute diverse GUI-Bugs gefixed. Keine Probleme.

02.12.2011

Heute war das fünfte Meeting: Sprint 3 abgeschlossen. Lief m.M. besser als der zweite. Sprint 3 geplant.

Vorschlag von Thomas: Bugliste sollte ersetzt werden (Excel ist umständlich). Bin derselben Meinung. Die GitHub Issues funktionieren besser als die Excel liste. Bugs werden nun als Issue mit dem Label „BUG“ erfasst. Habe div. Bugs gefixed.

05.12.2011

Habe heute GUI und Logik zu Reservierung stornieren implementiert. Keine Probleme.

07.12.2011

Mein Eindruck: Commitment bzwgl. laufender Aktualisierung des Sprint Backlogs durch Verwendung von GitHub nun viel besser.

09.12.2011

Heute war das sechste Meeting: Sprint 3 abgeschlossen und Sprint 4 geplant. Martin will das Product Backlog im GitHub Wiki abbilden. Ist von mir aus sinnvoll. Martin wird Zwischenpräsentation vorbereiten.

11.12.2011

Div. Programmierarbeiten durchgeführt.

14.12.2011

Heute war Zwischenpräsentation. Wir sind nicht dran gekommen. Wenn man sieht, was die anderen so präsentiert haben, dann ist man doch sehr beruhigt.

16.12.2011

Heute war das siebte Meeting: Wie immer: Sprint Review, Retrospektive und dann Planning von Sprint 5. Insgsamt seht guter Eindruck: Wir haben schon fast alle Anforderungen der LVA-Leitung umgesetzt.

18.12.2011

Bin im Rückblick positiv überrascht: Commitment zum Prozess ist immer noch hoch. Außerdem funktioniert das mit dem wöchentlichen Meeting besser als erwartet. Wir sind so gut wie immer vollzählich.

Habe letzte Arbeiten vor Weihnachten durchgeführt.

12.01.2012

Letztes Meeting: Sprint 6 als letzter Sprint. Mehr oder weniger nur noch Kleinigkeiten.

18.01.2012

Heut war Endpräsentation: Lief alles wunderbar.

Tagebuch – Thomas Perl

4.11.2011 - Meeting

Das erste Projekttre_en war sehr angenehm. Mein Vorschlag, Python und eine Web-GUI zu verwenden, wurde nicht angenommen. Stattdessen haben wir uns für Java und Swing entschieden - damit kann ich leben. Ansonsten bin ich froh, dass wir Git zur Versionskontrolle und LaTeX für die Dokumentation verwenden.

Die Aufgabenverteilung passt mir, freue mich schon, mit dem Team zu arbeiten.

11.11.2011 – Meeting

Nachdem wir in der ersten Woche mal grundlegende Sachen erledigt haben (und leider auch in anderen LVAs genug zu tun hatten), haben wir heute schon das zweite Treffen. Userstories sind gut angelegt, und ich werde in dieser Woche die Grundstruktur des Projektes anlegen.

14.11.2011 - Grundgerüst und GUI

Heute wurde das Grundgerüst für das Projekt erstellt - ein Java-Projekt mit dem Namen „BlueHotel\ und einer groben Oberfläche. Der Code wurde mit Vorausschau auf zukünftige Erweiterungen durch Projektmitglieder sehr offen gehalten, d.h. es wird mit möglichst abstrakten Konstrukten gearbeitet, die dann einfach durch Subclassing bzw. Generics konkretisiert werden können. Implementiert wurde unter anderem: Editor (ein Interface, das beschreibt, was ein Editor-Objekt mit Objekten anstellen kann), EditorManager (eine Factory, die zu einem Objekt von einem bestimmten Typ den richtigen Editor liefert) und ObjectList (eine grafische Liste von Objekten, die das Anlegen, Bearbeiten und Löschen von Objekten unterstützt, und sich dabei der vorher genannten Klassen bedient. Unzufrieden bin ich mit der technischen Unzulänglichkeit von Java und Swing - das, was in anderen Toolkits sehr leicht geht, ist in Swing sehr schwierig und mühsam. Stefan's

Tipp, „Window Builder Pro“ zu verwenden, hilft mir hier allerdings, denn mit diesem

Eclipse-Plugin kann man das Grundgerüst der GUI einfach zusammenklicken. Das macht den Code zwar nicht schöner, bringt aber viel schneller Ergebnisse, auf die man aufbauen kann. Gut, dass wir jemanden im Team haben, der sich mit diesen Tools auskennt!

18.11.2011 - Meeting

Ein weiteres wöchentliches Meeting hat heute stattgefunden - ich konnte bereits meine ersten Code-Ergebnisse präsentieren. Bei den Tests hängen wir momentan noch ein bisschen nach, aber ich bin der Meinung, dass es gut ist, als Basis für die Diskussionen einmal ein herzeigbares Projekt zu haben.Habe dem Projektteam erklärt, wie meiner Meinung nach das Editor-Interface zu funktionieren hat. Insgesamt stehe ich dem Projekt zuversichtlich gegenüber.

25.11.2011 - Meeting

Diese Woche ist von meiner Seite nicht viel weitergegangen, dafür habe ich aber schon für die nächste Woche einiges eingeplant. Das Erstellen von Rechnungen wird in den nächsten Tagen zu erledigen sein.

30.11.2011 – Eingabevalidierung

Ein Requirement, dass wir beim letzten Treffen besprochen haben, ist die Eingabevalidierung. Um den Aufwand wieder so gering wie möglich zu halten, wurde auch hier wieder sehr abstrakt gearbeitet - so wurde das Editor-Interface um Funktionen erweitert, die ein Objekt auf Vollständigkeit und Korrektheit überprüfen können. Im Fehlerfall gibt es auch eine Funktion, die aus einem Objekt die Fehler als von Menschen lesbaren Text ausgeben kann. Weil das Handling von Fehlern immer gleich ist (Validierung und wenn fehlgeschlagen, dann Fehlermeldung anzeigen, ansonsten fortfahren), wurde auch diese Logik in die Klasse „ValidationHandler\ gekapselt. Für den Kunden-Editor habe ich diese Funktionalität heute einmal komplett ausprogrammiert - das soll auch als Code-Beispiel für meine Gruppenmitglieder dienen, wenn diese die Funktionalität für ihre Module implementieren. Eine Beschreibung des Mechanismus habe ich per E-Mail ans Team geschickt. Aus der Validierung heraus ergeben sich einige fehlende Funktionalitäten, die ich als Bug-Reports erstellt habe. Momentan verwalten wir die Bugs als Excel- Dokument, was meiner Meinung nach suboptimal ist - die Github Issues eignen sich viel besser dafür. Ich habe das jetzt einmal per E-Mail deutlich zur Sprache gebracht, und werde auch beim nächsten Treffen versuchen, diese Änderung durchzubringen, denn auf die Dauer ist das Arbeiten mit dem Excel- Dokument sehr mühsam, und man hat schlecht Überblick über den zeitlichen Verlauf. In einem Bugtracker (wie Github Issues) sieht man schön, was noch offen ist, und wie sich der Status von Bugs geändert hat.

30.11.2011 - Rechnungs-Assistent

In der zweiten Programmier-Session des heutigen Tages habe ich den Rechnungs- Assistenten implementiert. Dieser hilft dabei, eine Liste von Reservierungen anzuzeigen, und diese dann zu verbuchen. Das wird momentan einfach in der GUI angezeigt, und ist noch nicht ausimplementiert.

Auch hier stoße ich entweder auf fehlendes Swing-Wissen meinerseits oder auf Limitierungen von Swing - so musste ich für die Liste der Reservierungen (eine Multi-Selektions-Liste) eine eigene Klasse erstellen. Es funktioniert, aber macht den Code etwas unübersichtlicher.

01.12.2011 - Einbauen von Invoices

Heute hatte ich nur wenig Zeit, am Projekt weiter zu arbeiten. Ich habe ein Icon für die Rechnungen zum Projekt erstellt, um diese in der UI sichtbar zu machen.

02.12.2011 - Meeting

Ich habe meine Bedenken über die Führung der Bug-Liste als Excel-Dokument dem Projektteam bekanntgegeben. Nach einiger Diskussion haben wir uns entschieden, die Bugs wie vorgeschlagen nach Github zu übertragen. Diese

Aufgabe übernehme ich gerne, da es meine Arbeit im Projekt in Zukunft erleichtern wird.

02.12.2011 - Bugs zu Issues konvertiert

Nach dem letzten Meeting wurde beschlosen, dass die Excel-Liste für die Bugs endlich wegkommt, und wir stattdessen Github Issues verwenden. Darüber bin ich sehr froh, und es wird meine Motivation für das Projekt steigern. Weiters haben wir uns auch entschlossen, das Product-Backlog nicht mehr als LaTeX-Datei zu führen, sondern ins Github-Wiki zu übernehmen.Beide Änderungen habe ich heute gemacht, und per E-Mail das Team informiert.

09.12.2011 – Meeting

Dieses Meeting war recht kurz, wir haben vorallem in Hinblick auf die bevorstehende Präsentation schon einige Punkte besprochen. Allgemein ist zu sagen, dass wir ca. ein Drittel des Projekts fertig haben, vielleicht sogar ein bisschen mehr. Insgesamt habe ich das Gefühl, dass wir gut in der Zeit liegen, und gut miteinander auskommen. Vorallem die wöchentlichen Treffen am Freitag sind ein guter Abschluss der Uni-Woche, und geben uns Ansporn, in der nächsten Woche am Projekt weiter zu arbeiten.

09.12.2011 – Fehlerüberprüfung beim Löschen

Beim Löschen von Kunden und Zimmern ist darauf zu achten, dass diese nicht gelöscht werden dürfen, wenn sie in Reservierungen vorkommen. Dies wurde beim letzten Meeting besprochen, und ich habe das jetzt implementiert - wobei nun nur ein Fehler angezeigt wird. Idealerweise sollte hier noch eine bessere Meldung erscheinen - werde das beim nächsten Meeting zur Sprache bringen. Seit den letzten 2 Wochen geht es wieder gut voran beim Projekt, die Meetings jeden Freitag helfen, das Projekt am Laufen zu halten - selbst wenn wir oft nur eine halbe Stunde oder Stunde über das Projekt sprechen. Der wöchentliche Austausch ist für so ein Studentenprojekt meiner Meinung nach sehr wichtig. Ein Online-Treffen (Textchat oder VoIP) würde wohl nicht so viel Motivation bringen. Weiters habe ich heute Rechnungs-Infos zum Datenmodell hinzugefügt, und einige Clean-Ups durchgeführt.

13.12.2011 - Lösch-Checks in der Objektliste

Heute habe ich die Issue 26 behoben - in der Objekt-Liste wird nun immer eine Bestätigung des Löschens angezeigt (bzw wenn nicht möglich, dann eine andere Meldung).

14.12.2011 - Präsentation

Für die Präsentation habe ich heute die Präsentation (Demo) mit aktuellem Stand abgelegt. Bin recht zufrieden mit dem aktuellen Zwischenstand, die Stimmung im Projekt ist gut - es ist schön, einen kleinen Meilenstein erreicht zu haben.

16.12.2011 - Meeting

Heute haben wir vorallem zum Thema Rechnungslegung eine offene Punkte besprochen. Ich hoffe, diese kann ich bald umsetzen. Einige Bugs, die dafür offen

sind, werden von den Projektmitgliedern hoffentlich bald erledigt. Jetzt haben wir noch Zeit, um einige Nice-to-have Features umzusetzen. Die GUI ist in der Zwischenzeit schon recht angewachsen, ich hoffe, dass wir am Ende des Projekts noch etwas Zeit finden, um die GUI etwas aufzuräumen.

14.01.2012 - Rechnungs-Generierung

Heute habe ich den Code für die Erstellung der Rechnungen implementiert. Die Implementierung wird von den Projektmitgliedern noch getestet werden. Auch hier war einiges schwerer als erwartet, aber im Ende_ekt funktioniert es jetzt rudimentär sehr gut. Was hier noch zu erledigen ist, ist etwas Polishing für die Rechnungen und ein paar Verbesserungen bei der Berechnung. Das sind aber alles Sachen, die dann später erledigt werden können. Als „Feature“ kann das Rechnung Erstellen nun abgehakt werden.Das Projekt geht dem Ende zu - es sind nur mehr ein paar Kleinigkeiten zu erledigen.Wenn es sich ausgeht, will ich mir vor der Endpräsentation noch ein klein wenig Zeit nehmen, um die GUI und die Rechnungen noch schöner zu machen.

17.01.2012 - Polishing

Heute habe ich fast den ganzen Tag mit Polishing der UI verbracht. Ich bin ein wenig ausgelaugt, deshalb kein detaillierter Report heute, sondern nur ein kleiner Überblick über die Änderungen: Suchen/filtern in der Objektliste, mehr Spacing/Padding in der GUI, Raumbelegungs-Platz als eigenen Menüpunkt in der Menüleiste, korrigieren der Preis-Eingabe im Zimmer-Editor, Rechnungslegung verbessert und gepolished. Ich freue mich schon auf die Präsentation, und _nde, dass das Programm mit den heutigen Änderungen schon sehr professionell aussieht. Habe die Änderungen den Projektmitgliedern mitgeteilt.

18.01.2012 - Endpräsentation

Heute war die Endpräsentation. Durch das Polishing gestern ist das Endprodukt heute sehr schön, und Alexander hat sich darum gekümmern, Test-Daten für die Demo zu erstellen. Freue mich schon auf den Abschluss des Projekts!

20.01.2012 - Schreiben des Endreports

Da ich in der kommenden Woche Uni-Sachen zu erledigen habe, und am Wochenende danach nicht in Wien bin, habe ich heute meinen Teil des Endreports erledigt. Rückblickend bin ich sehr zufrieden mit dem Ergebnis, und finde, dass wir ein herzeigbares und schönes Projekt haben, aber gleichzeitig auch einiges in Sachen Java und Projektmanagement gelernt haben. Das Arbeiten im Team war gut, und vorallem die wöchentlichen Meetings steigerten die Motivation.

Tagebuch – Martin Wieser

Meeting - 04.11.11

Heute gab es das erste Meeting um 12:00 uhr. Es wurde schon viel geklärt:

- repo: github
- docu: latex (texworks)
- sprache: java

- datenbank: hsqldb
- gui: swing

und Aufgaben verteilt:

Rollen:

- SM - Stefan
- PO - Martin
- Team - Ale, Thomas

TODO:

- alle: SCRUM vertraut machen, besonders mit seiner Rolle
- 2-3 Userstories schreiben
- persönliches tagebuch schreiben
- Thomas & Ale: Projekt einrichten, libs, entwurf für architektur
- Stefan: Report (wie soll er ausschauen?), nächste Meeting organisieren
- Martin: Github einrichten, Latex template

Ich denke die Aufgaben sind alle leicht schaffbar. Das Team ist noch sehr motiviert und ich sehe noch keine Probleme.

Initialisierung - 05.11.11

Habe heute meine Aufgaben zum größten Teil bereits erledigt. Werde mich in den nächsten Tagen weiter mit meiner Rolle im Projekt vertraut machen. Nächstes Gruppenmeeting wurde auf Freitag festgelegt. Bis dahin ist noch viel Zeit und die Aufgaben sind leicht schaffbar.

Zweites Meeting - 11.11.11

Zunächst wurde von jedem Mitglied Bericht über die vergangene Woche erstattet. Daraus resultierte, dass alle Aufgaben erledigt wurden. Anschließend wurde eine neue Technologie für die GUI-Erstellung vorgeschlagen und angenommen. Im SP1 wurden nachher vom PO einige Userstories erklärt und der SM und das Team haben einige Userstories in den Sprint 1 übernommen:

- Kunde anlegen
- Zimmer anlegen
- Reservierung vornehmen
- Daten löschen
- Daten bearbeiten

wobei die Aufgabe folgend verteilt wurden:

- Thomas: Grundstruktur aufbauen + CRUD für Kunde
- Stefan: Sprint Backlog + BDS
- Ale: CRUD für Reservierung
- Martin: Product Backlog anpassen + CRUD für Zimmer

Das Projekt nimmt schön langsam gute Züge an, sollten alle Aufgaben bis zum Ende des Sprints (nächsten Freitag) erledigt sein. Sind wir auf einem sehr guten

Weg. Die Architektur des Teams ist sehr vielversprechend. Die eingebundenen Framework werden uns viel Arbeit ersparen.

Anpassung Product Backlog - 12.11.11

Den Product Backlog habe ich um die besprochenen Userstories erweitert. Der PB schwellt so weiter an und das wird immer mehr werden. Ich muss darauf achten, dass das Team sich nicht übernimmt und nicht in Details verrennt.

Erstes Programmieren - 15.11.11

Das gesamte Team geht mit richtig Schwung an die Sache. Es wurde der Sprint Backlog fertig gestellt, ein Konzept für das Testen vorgeschlagen, ein Template für GUI und DAO aufgesetzt. Außerdem wurde bereits ein Datenbank eingerichtet sowie Kunden und Zimmer darin erfasst. Windows Builder Pro, ein neues Tool, hängt sich hin und wieder auf, aber funktioniert ansonsten recht gut. Vorallem spart man sich die ganze Schreiberei bei der GUI. Ich sehe für diesen Sprint und für das gesamte Projekt keinerlei Probleme, im Gegenteil, es scheint als wären wir zu schnell.

Drittes Meeting - 18.11.11

Beim Meeting wurde zunächst die Aufgaben der letzten Woche besprochen und der erste Sprint abgenommen. Anschließend wurde folgende Aufgabe verteilt:

Stefan: GUI, Template für Testfälle + Bugliste, Funktionale Tests für Kunde

Alexander: Neue Tabelle => mehrere Zimmer reservieren, Auslagern der Logik

Thomas: Unittests für Logik, Funktionale Tests für Reservierung + Zimmer

Martin: Product Backlog adaptieren

Die Tests aus Sprint 1 sind noch offen, trotzdem war die Bestandteile stabil und konnten abgenommen werden. Zukünftig werden Tests erstellt. J-Unit und Funktionale.

Das Projekt schreitet sehr schnell voran. Der zweite Sprint ist geprägt von Überarbeitung der GUI sowie Testanpassung. Trotzdem sind auch Fortschritte geplant.

Tests - 22.11.11

Zur Zeit ist etwas der Schwung raus. Nach anfänglicher Euphorie geht es nun etwas langsamer voran, da einige Bugs aufgetreten sind. Die Behebung dieser Fehler wird im nächsten Sprint viel Zeit in Anspruch nehmen. Außerdem müssen wir die Test ausbauen, damit wir Fehler schneller erkennen und dadurch nicht mehr derartig viel nacharbeiten müssen.

Viertes Meeting - 25.11.11

Nach der Besprechung der vergangenen Wochen liegen einige offene Baustellen vor uns. Diese werden neu verteilt:

Stefan: GUI, Reservierung stornieren

Alexander: Logik

Thomas: Rechnung erstellen, Frühzeitige Abreise

Martin: Unittests für Logik, Funktionale Tests für Reservierung + Zimmer

Hoffentlich verläuft der kommende Sprint zügiger und sauberer durch. Sollte dies allerdings der Fall sein, schauen wir in eine rosige Zukunft.

Testen - 28.11.11

Wollte heute meine Tests einfügen, allerdings fehlen einige libs, was eine kleine Verzögerung mit sich bringt. Ärgerlich...

Testen - 29.11.11

Nach einer kurzen Email wurden alle Mängel schnell behoben. Die Test konnte ich dann zügig durchführen und einige Fehler erkennen, welche ich teils selber ausbessern konnte. Andere Bugs habe ich an die jeweiligen Ersteller gesendet.

Werden die erkannten Fehler behoben, liegt vor uns eine äußerst stabile Software. Durch die eingeführten Tests wurde auf jeden Fall die Qualität deutlich gesteigert.

Meeting -02.12.11

Zunächst hat jeder seine bisherige Arbeit kurz beschrieben, wobei einige Arbeiten durch andere blockiert wurde. Außerdem wurde die doppelte Buchführung von Bugs bemängelt.

Dadurch wurde beschlossen von nun an alle Aufgabe und Bugs im GitHub zu dokumentieren.

Dies soll schnelleres Arbeiten auf Grund der schnelleren Kommunikation ermöglichen.

Für die kommende Woche wurde alle Issues in GitHub eingetragen und verteilt. Somit werden einige Bugs ausgebessert die Rechnungen vervollständigt und die GUI sowie DAO überarbeitet.

DAO - 06.12.11

Durch die ausschließliche Doku in Github wird die Arbeit deutlich leichter. Die zugewiesenen Issues werden abgearbeitet und geschlossen. Den Rest (Emails versenden, ...) erledigt das System.

Die DAO konnte ohne Probleme erweitert werden, sobald die ``Zauberei'' von Hibernate verstanden wurde.

Meeting -09.12.11

Die geforderten Aufgaben wurde vollständig erfüllt und das System ist lauffähig. Das Team ist weiterhin sehr zuversichtlich. Die Änderungen bezüglich der Dokumentation haben sich positiv ausgewirkt, wodurch die Arbeiten erleichtert

wurden. Eine Überarbeitung der GUI sowie einige Funktionen, welche die Usability betreffen sollen eingearbeitet werden.

Meeting - 16.12.11

Das Ausdrucken der Rechnung muss überarbeitet werden, um allen Anforderungen gerecht zu werden. Ansonsten wurden alle Aufgabe zufriedenstellend abgearbeitet. Aufgrund der kommenden Ferien werden nun größere Aufgaben vergeben, wodurch das Projekt auch zum Abschluss gebracht werden soll.

Zimmerbelegung anzeigen:

Als Geschäftsführer will ich in die Zimmerbelegung einsehen.

Suchfunktion in Anzeige:

Als Rezeptionist will ich über eine Suche schnell zu gewünschten Kunden, Zimmern oder Reservierungen gelangen.

Öffnen des Bearbeiten-Menüs nach Doppelklick:

Als Rezeptionist will ich durch einen Doppelklick auf einen Datensatz, diesen bearbeiten.

Meeting -12.01.11

Alle must-have- und einige nice-to-have-Tasks wurde abgearbeitet. Nun soll das Produkt

für die kommende Präsentation vorbereitet werden. Das gesamte Team ist mit dem Endprodukt recht zufrieden.

Endpräsentation -18.01.11

Die Vorstellung des Projektes war ein voller Erfolg. Das Produkt hat ohne Probleme funktioniert.