www.scrum.org/Resources/What-is-Scaled-Scrum

Wenn viel nicht einfach viel hilft: Warum Skalieren mehr ist als Organisieren

Zahllose IT-Abteilungen und -Organisationen haben in den letzten Jahren erfolgreich Projekte mit Scrum organisiert. Daran waren meistens ein oder zwei Teams beteiligt. Die positiven Erfahrungen führen scheinbar automatisch zum nächsten Schritt – großen Projekten mit mehreren Scrum-Teams. Denn wenn die Produktivität mit einem Scrum-Team steigt, dann – so die Hoffnung – wird sie das bei mehreren Teams erst recht. Folglich wird die Time-to-Market sinken und die Wettbewerbsfähigkeit gestärkt. Leider bleibt die Hoffnung oftmals unerfüllt. Viele Unternehmen skalieren zu früh, zu schnell und zu unbedacht. Dieser Artikel analysiert, wann sich Skalieren lohnt, und beschreibt typische Skalierungsfallen. Außerdem stellt er mit „Scrum.orgs Nexus" ein Framework vor, das dabei hilft, professionelles Scrum zu skalieren und nicht einfach mehrere Teams zu organisieren.

Scrum zu skalieren bedeutet, dass mehrere Teams im gleichen Projekt in einem oder mehreren Sprints zusammenarbeiten, um am Ende eines Sprints ein gemeinsames fertiges Inkrement zu liefern, ein „Integrated Done Increment". Es ist praktisch immer der gleiche Grund, der Unternehmen dazu bewegt, dieses Vorgehen als Mittel der Wahl zu betrachten: der Wunsch nach steigender Produktivität. Dabei hält sich hartnäckig die Überzeugung, wonach sich große Systeme nur mit vielen Leuten entwickeln lassen. Die Annahme, die dahinter steht, ist einfach: Wenn ein Scrum-Team für ein Produkt eine bestimmte Zeit braucht, dann ist das gleiche Produkt bei vier Teams in einem Viertel der Zeit fertig. Dabei können es auch drei, neun oder eine andere Zahl von Teams sein, die Formel bleibt gleich: Mehr Teams steigern die Produktivität proportional.

Die Erfahrung zeigt, dass diese verlockend simple Formel einen Nachteil hat: Sie trifft meistens nicht zu. Denn mit der Skalierung steigen die Komplexität und die Zahl der Abhängigkeiten zwischen den Teams. Beides senkt die erhoffte Produktivität und gefährdet die gemeinsame Lieferung im Sprint. Es sind häufig mehrere Fallen, die zwischen der Entscheidung für die Skalierung und dem angestrebten Erfolg lauern. Jede einzelne hat das Potenzial, den ganzen Ansatz zum Scheitern zu bringen.

Erste Skalierungsfalle: verfrühter Zeitpunkt

Zu früh gewählt ist der Zeitpunkt zum Skalieren dann, wenn sich die Beteiligten noch in einem unprofessionellen Stadium oder Umfeld befinden. Denn in diesem Fall steigt die Komplexität in einer nichtlinearen Form. Aus kleineren Fehlern oder Schwachstellen werden massive Hindernisse. Anders gesagt: Bevor man daran gehen kann, Scrum zu skalieren, muss Scrum grundsätzlich professionell etabliert sein. Das klingt selbstverständlich, ist es in der Praxis jedoch oft nicht. Schwache Professionalität kann sich in jedem der in **Abbildung 1** illustrierten Teilaspekte „Qualität & Lieferfähigkeit", „Value" und „Prozess" zeigen. Lassen Sie uns diese drei Aspekte einzeln betrachten.

Abb. 1: Die drei Teilaspekte der Agilität.

Quality & Delivery

Schlechte Qualität und manuelles Deployment verschlingen in IT-Projekten oft einen Großteil der Produktivität und des Budgets. Im Projektverlauf häufen sich immer mehr technische Schulden an. Sie führen letztendlich dazu, dass immer weniger Zeit und Kapazität frei bleiben, um neue Features zu entwickeln. Deshalb ist es so wichtig, die Prinzipien aus dem *eXtreme Programming (XP)* anzuwenden, die handwerklichen Fertigkeiten des Agile-Software-Engineerings zu trainieren und Tests weitgehend zu automatisieren. Guter Code ist und bleibt die Basis guter Software, unabhängig von der Organisationsform eines Projekts.

Value – Produktmanagement

Engineering-Fertigkeiten sind für den Aspekt der Lieferfähigkeit fundamental, garantieren aber noch nicht den „Value" des Produkts. Diesen Wert sollen, in der Theorie, die Auswahl effektiver Produktstrategien sowie die Priorisierung der Features sichern: Der Product Owner wählt pro Sprint diejenigen Features aus, die wirklichen Nutzen bringen. In der Praxis wachsen Product Backlogs leider und fälschlicherweise sehr schnell und werden sehr lang. Die falschen Features herauszugreifen, ist da ebenso leicht wie riskant, weil bei mehreren Teams in jedem einzelnen Sprint viel Geld auf dem Spiel steht. Abhilfe kann das Konzept des *Minimal Viable Products (MVP)* (vgl. [Rie09]) aus der Lean-Start-Up-Philosophie schaffen: Der Product Owner wählt nur diejenigen Features aus, die minimal erforderlich sind, um ein erstes Produkt mit dem angestrebten Nutzen zu erhalten.

Das setzt allerdings voraus, dass der Product Owner diejenige Instanz ist, bei der sich das gesamte Wissen über das Produkt, alle Anforderungen und alle strategischen

Ziele kanalisieren. Außerdem muss er oder sie die zentrale Entscheidungsmacht haben, was das Produkt anbelangt.
In vielen realen Projekten hat der Product Owner dagegen eine Placebo-Funktion, tatsächlich gibt es dann endlose, verwässerte Entscheidungsprozesse quer durch alle Ebenen und Bereiche hindurch. Aus „alle entscheiden“ wird dann schnell „niemand entscheidet“, schon gar nicht in der Geschwindigkeit, die agile Vorgehensweisen brauchen. Deshalb sind starke, entscheidungsbefugte und -fähige Product Owner eine der Voraussetzungen für professionelles Scrum.

Continous Improvement
Für den Change-Prozess bzw. die Transition ist die Beteiligung des Managements unerlässlich. Mit langsamen, trägen Veränderungsprozessen lassen die Erfolge auf sich warten. Schwache Scrum Master ohne wirklichen Einfluss können die Agilisierung nicht vorantreiben. Der von Scrum vorgesehene kontinuierliche Verbesserungsprozess bedarf eines konsequenten, strukturierten Change- und Transition-Managements – und eines klaren Bekenntnisses der Unternehmensführung.

Zweite Skalierungsfalle: zu hohes Tempo

Das Bekenntnis des Managements zur Transition ist da, die Skalierung wird vorangetrieben – und hier lauert die nächste Falle. Zu schnell zu viele Teams einzubeziehen, birgt die Gefahr, keine Basis zu haben, von der aus die wachsende Komplexität gemeistert werden könnte. Basis bedeutet hier vielerlei: In allen bereits beteiligten Teams sollte das Wissen über die fachlichen Hintergründe, die Technologie und die Prozesse vorhanden sein. Außerdem muss die Einstellung zur Kooperation passen. Ebenso wichtig ist eine Systembasis, die mehrere Teams weiterentwickeln können. Ohne diese Basis ist ein sehr hoher Koordinationsaufwand erforderlich, der oft unterschätzt wird.
Organisch vorzugehen, senkt den Aufwand und erhöht erfahrungsgemäß die Erfolgswahrscheinlichkeit. Wenn ein Team die agilen Prinzipien selbstverständlich lebt, können seine Mitglieder das erworbene Wissen fachlich, prozessbezogen und technisch in ihre neuen Teams weitergeben. Über diese teaminternen Coachs baut sich gemeinsames Wissen auf, mit dem mehrere Teams arbeiten können.
Zunächst reicht es, ein weiteres Team einzubeziehen. Erst wenn das kontinuierliche „Inspect&Adapt“ zeigt, dass beide neuen Teams das Niveau professionellen Scrums erreicht haben, stoßen sukzessive weitere Teams dazu (**siehe Abbildung 2**). Diese Art des organischen Wachstums stellt sicher, dass in jedem neuen Team mindestens ein erfahrener Experte vertreten ist.

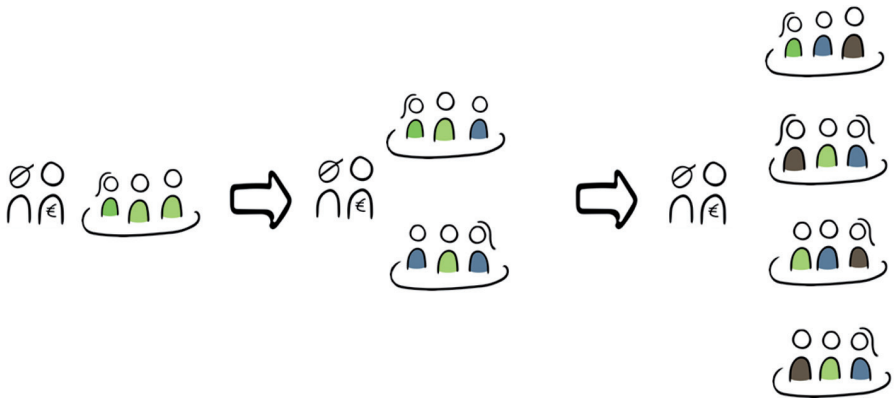

Abb. 2: Kontrollieren statt übereilen: Inspect&Adapt und organisches Wachstum.

Dritte Skalierungsfalle: Vernachlässigung der Skalierungskosten

Skalierung hat auf jeden Fall einen Preis. Mit professionellem und organischem Vorgehen lässt er sich deutlich senken, grundsätzlich vorhanden ist er dennoch. Denn wenn mehr als ein Scrum-Team an einem Product Backlog arbeitet, dann steigt die Zahl an Interaktionen und nicht-linearen Ereignissen stark an und mit ihnen die Komplexität. Das verursacht Kosten. Kommunikation, Synchronisation und Organisation bewirken einen Skalierungs-Overhead, der oft unterschätzt und nicht ins Budget eingerechnet wird. Das gilt umso mehr, als die Korrelation zwischen der Anzahl der Scrum-Teams, dem Kostenanstieg und der Steigerung der Produktivität nicht linear ist. Bevor ein Product Owner die Entscheidung fällt, mehrere Scrum-Teams einzubeziehen, sollte er die Vorteile sorgfältig gegen die zusätzlichen Kosten abwägen.
Damit wird deutlich, dass Skalierung zwar ein Weg sein kann, die Produktivität zu steigern, das einzige Mittel ist sie sicher nicht. Im Gegenteil empfiehlt es sich, zuerst die Professionalität zu erhöhen, was sich bereits positiv auf die Produktivität auswirkt. Skalierung sollte eher als letztes Mittel in Betracht gezogen werden.
Aber was können Unternehmen tun, die sich bereits für die Skalierung entschieden haben, in eine der drei Fallen geraten sind und jetzt nicht die gewünschten Ergebnisse erzielen?

Wenn Skalierung nicht funktioniert

Dass die Skalierung nicht funktioniert, lässt sich einfach daran festmachen, dass am Ende eines Sprints kein integriertes, auslieferbares Inkrement (*integrated done increment*) aller Teams geliefert wird. Demnach sind die Probleme zu tiefgreifend, als dass sie sich mit einer lokalen Optimierung lösen ließen. Erforderlich sind systematische Veränderungen, die ein strukturiertes und kontinuierliches Vorgehen verlangen. Das Unternehmen sollte umgehend ein Change-Team mit Beteiligung des Managements gründen. Letztere ist sehr wichtig, um die Entscheidungsfähigkeit des Teams sicherzustellen. Dieses Change-Team analysiert die Schwachstellen einerseits subjektiv, etwa durch Interviews mit Betroffenen. Andererseits kann es mittels Assessments die aktuelle Situation objektiv erfassen. Aus den Analysen leitet das Change-Team dann Maßnahmen ab, die darauf zielen, die Professionalität systematisch zu erhöhen. Den Erfolg dieser Maßnahmen misst das Team kontinuierlich mittels ausgewählter Metriken. Im ersten Schritt verläuft diese kontrollierte, strukturierte Professionalisierung parallel zur eigentlichen Skalierung: An der Zahl der kooperierenden Teams verändert sich erst einmal nichts.
In manchen Projekten reicht das nicht aus, um die Probleme in den Griff zu bekommen. Das ist der Zeitpunkt, um zu de-skalieren. De-Skalierung bedeutet, die Zahl der beteiligten Teams zu senken und damit die verbundene Komplexität zu reduzieren, um sie wieder steuerbar zu machen. Mit der geringeren Team-Anzahl ist es dann eher möglich, sich parallel der Professionalisierung zu widmen.
Wenn selbst dieses Vorgehen nicht mehr greift, hilft nur noch das, was Ken Schwaber als „Scrumble“ bezeichnet: Ein tempo-

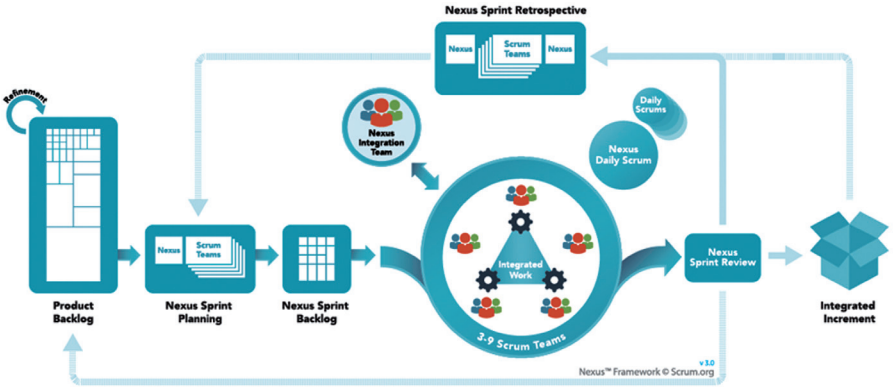

Abb. 3: Nexus, schematische Darstellung.

rärer Stopp des Projekts, in dem erst einmal gar keine Features mehr entwickelt werden und de-skaliert wird. In dieser Phase beginnen tiefgreifende und umfangreiche „Aufräumarbeiten", die ganze Konzentration liegt auf der Professionalisierung.

Die beiden Optionen des De-Skalierens und des „Scrumble" klingen unschön. Die Erfahrung lehrt jedoch, dass in Schieflage geratene Skalierungsprojekte kaum andere Möglichkeiten haben, wieder auf den Pfad der Tugend zurückzukommen. Denn wenn die Beteiligten nicht selbst die Entscheidung fällen, beispielsweise die Zahl der Teams zu reduzieren, dann tut es meistens das Management.

Deshalb sollten diejenigen, die sich für eine Skalierung entscheiden, zwei Dinge idealerweise von Anfang an beachten: Die ständige Professionalisierung und die Wahl eines leichtgewichtigen Skalierungsprozesses. Dazu hat Scrum.org das Skalierungs-Framework „Nexus" entwickelt.

Nexus: der Hintergrund

Ein Nexus ist grundsätzlich eine kausale Verbindung von Dingen oder Menschen. Bei Scrum.org ist Nexus ein Framework,

Interview mit Ken Schwaber zu Nexus

Warum Nexus?

Das Wort Nexus bezeichnet eine Gruppe oder Folge, deren einzelne Mitglieder bzw. Teile integriert, also miteinander verbunden sind. Wenn ich das Wort „Nexus" verwende, dann meine ich damit auch, dass diese Gruppe oder Folge gemeinsam handelt und als Einheit betrachtet wird. Nexus addiert genau so viel zum Scrum-Framework, dass diese Vorstellung real wird. Es liefert die nötigen Tanzschritte, um als integriertes Ganzes zu handeln. Die Fähigkeit zu tanzen – oder zu integrieren – ist dann immer noch eine schmerzhafte Herausforderung. Sie erfordert Training und kontinuierliche Verbesserung, bis sie in Fleisch und Blut übergegangen ist.

Abhängigkeiten zu identifizieren und zu entfernen sowie eine kontinuierliche Integration, um die aktuelle Qualität zu bestimmen, sind die minimalen Anforderungen. Dennoch erwarte ich, dass viele nicht die Ideen des De-Skalierens und Scrumble aufgreifen werden. Organisationen sollten die Kunst des Machbaren schätzen, wenn sie skalieren. Sie sollten sich bewusst sein, eher zu de-skalieren als weiter technische Schulden zu generieren.

Was ist so besonders an Nexus?

Nexus ist eine schlanke Scrum-Erweiterung, ein äußeres Skelett. Die eigenen Artefakte der einzelnen Scrum-Teams werden integriert in die Nexus-Artefakte (Nexus Sprint Backlog, Integrated Increment). Events, um die Integration zu managen, indem Abhängigkeiten erkannt und gelöst werden, lenken die existierenden Scrum-Events, beispielsweise Nexus Print Planning, Nexus Daily Scrum und Nexus Sprint Retrospective. Wo es angebracht ist, ersetzen sie diese auch (Nexus Sprint Review). Anstatt auf glückliche Zufälle zu hoffen, wird die Verantwortung für die Integration in eine neue Rolle eingebettet, das Nexus Integration Team.

Nexus ist schlank, moduliert das existierende Scrum-Framework und baut konsistent auf praktischem Scrum-Wissen auf, das Scrum-Anwender weltweit bereits erworben haben.

Nexus+[1]) bietet eine architektonische Plattform, um die Arbeit zahlreicher Scrum-Teams in ein konsistentes Ganzes von hoher Qualität zu integrieren. Das ist entscheidend geworden, weil Softwareentwicklung mittlerweile einen großen Teil von geschäftskritischen und lebenswichtigen Produkten ausmacht. Nexus+ startet mit einfacheren Mechanismen wie Produktfamilien und COM-Architekturen, baut aber auf Platform Computing, um den hohen Entwicklungsstand und die Komplexität der Zukunft zu liefern.

Warum jetzt?

Der Hauptteil meiner Arbeit war mit denkenden Unternehmen. Der Dreh- und Angelpunkt ihres Erfolgs ist die Fähigkeit, innovative Software zu entwickeln und in Betrieb zu setzen.

*Innerhalb der Methapher des „Crossing the Chasm" hat das Bestreben nach Agilität jetzt die frühe und vielleicht die späte Mehrheit (**Early bzw. late majority**) erreicht. Wie zu erwarten, haben große IT-Organisationen Methoden und Tools gekauft, die ihnen den Königsweg zur Agilität bieten. Scrum wird oft in diesen Methoden gebündelt in der „Entwicklungsphase".*

Unglücklicherweise enthalten diese Methoden keine Anleitungen dazu, Softwareentwicklung und Scrum zu skalieren. Hier kommt Nexus ins Spiel, ein Framework, das über die Entwicklungsphase dieser Methoden gelegt werden kann. Nexus bringt außerdem eine konsistente Terminologie, mittels derer diese Organisationen in die Scrum-Community und -Diskussion eintreten können.

1) Anmerkung: Nexus ist für Entwicklungsgruppen von bis zu 100 Personen konzipiert, Nexus+ für größere Initiativen.

Kasten 1: Das Interview basiert auf einem E-Mail-Austausch zwischen Fahd Al-Fatish und Ken Schwaber vom 10. September 2015 (vgl. `https://kenschwaber.wordpress.com/2015/09/14/nexus/`*).*

das die bekannten Scrum-Prinzipien erweitert auf Projekte mit bis zu neun Entwicklerteams. Professionelles Scrum sieht vor, am Ende eines jeden Sprints lauffähige Software (ein „Done Increment“) zu liefern. Das ist bereits mit einem Team eine Herausforderung und bedingt eine lange Reihe von Voraussetzungen: Disziplin, Entschlossenheit, Fokussierung auf Entwicklungstechniken, Kommunikation, Zusammenarbeit und nicht zuletzt die Möglichkeit, Hindernisse aus dem Weg zu räumen, die Fortschritt und Kreativität hemmen.

Ist diese Liste schon beachtlich, so steigt mit der Skalierung der erforderliche Grad an Kooperation, Integration und Reife der eingesetzten Werkzeuge bzw. Techniken. Deshalb ist Nexus Scrum, geht aber auch darüber hinaus (eine schematische Darstellung des Nexus-Frameworks zeigt **Abbildung 3**). Genau wie bei Scrum gibt es weiterhin ein Product Backlog, einen Scrum Master und einen Product Owner. Genauso bleibt das Ziel, am Ende des Sprints ein „Integrated Done Increment“ abzuliefern – nur dass jetzt mehr als ein Team an diesem Inkrement arbeitet. Analog zu Scrum bezieht sich „Done“ nicht auf die Menge der abgeschlossenen Features pro Sprint, sondern darauf, dass die Features fertig im Sinne von „qualitativ auslieferbar“ sind.

Nexus zielt darauf, Abhängigkeiten zu lösen, die zwischen mehreren Teams auftreten. Anders gesagt besteht das Herz von Nexus daraus, die Arbeit der einzelnen Teams zu integrieren. Der Grad dieser Integration wird jeden Tag überprüft und liefert die Inhalte für die tägliche Planung der Scrum-Teams.

Unterstützung erhalten die Entwicklerteams dabei von einer Nexus-Besonderheit: dem Nexus-Integrationsteam. Dieses trägt die Verantwortung dafür, dass am Ende des Sprints ein „Integrated Done Increment“ geliefert wird. Dazu übernimmt es eine ganze Liste von Aufgaben, die bei der Koordination mehrerer Teams auftreten und die sich nicht einfach von selbst lösen. Beispielsweise analysiert das Team Abhängigkeiten im Sprint und macht sie transparent. Es stellt sicher, dass die richtigen Integrationspraktiken benutzt werden, und coacht die Entwicklerteams bei Bedarf im Agile Engineering. Es unterstützt bei der Auswahl einer geeigneten Architektur und arbeitet manchmal direkt an der Entwicklung der Features mit.

Entscheidend ist, dass Nexus – genau wie Scrum – ein Framework darstellt und kein Universalrezept. Wie Scrum baut es auf Selbstorganisation, Wissensaustausch und empirischen Erkenntnissen. Genau wie Scrum kann Nexus beliebig von allen genutzt werden. Um damit zu starten, bedarf es solider Scrum-Kenntnisse und eines Überblicks über Nexus selbst. Dafür steht der Nexus Guide zur Verfügung, der bereits in diverse Sprachen übersetzt wurde (vgl. [Scr16]).

Jeder Einsatz von Nexus beruht darauf, die für den jeweiligen Kontext geeigneten Vorgehensweisen zu wählen und anzuwenden. Manche Instanzen von Nexus funktionieren sehr gut in einem Projekt, dennoch erscheint es im nächsten Projekt ratsamer, andere auszuwählen. Deshalb folgt Nexus dem Grundsatz, die Ziele und den äußeren Rahmen festzulegen – etwa die Lieferung eines gemeinsamen Inkrements am Ende des Sprints –, für den Weg dorthin jedoch einen ganzen Kanon von Praktiken anzubieten. Etwas anderes käme eben doch dem Versuch eines Universalrezepts gleich. Im Interview erläutert Ken Schwaber die Hintergründe zur Entstehung von Nexus und was dieses Framework ausmacht (**siehe Kasten 1**).

Fazit

Gerade weil es für die Skalierung kein universelles Rezept gibt, profitieren viele Unternehmen direkter, wenn sie ihre Softwareentwicklung weiter professionalisieren. Allein dadurch steigt die Produktivität bereits. Außerdem ist diese hohe Professionalität ohnehin nötig, um erfolgreich zu skalieren. Sollte der Anstieg der Produktivität dennoch nicht ausreichen, ist es entscheidend, unverfälschtes, professionelles Scrum zu skalieren, anstatt einfach mehrere Teams zu organisieren. Mit Nexus hat Scrum.org dazu ein leichtgewichtiges Rahmenwerk vorgestellt.

Ungeachtet des gewählten Frameworks bedeutet Scrum zu skalieren, die Scrum-Werte wie Risikokontrolle, Transparenz und Kreativität auf eine größere Anzahl an Beteiligten zu übertragen. Es bedeutet nicht, sie zu vernachlässigen. ||

Links

[Rie09] E. Ries, Startup lessons learned, 2009, siehe:
`http://www.startuplessonslearned.com/2009/08/minimum-viable-product-guide.html`
[Scr16] Scrum.org, Download The Nexus Guide, 2016, siehe:
`www.scrum.org/Resources/The-Nexus-Guide/Downloads`

Der Autor

|| Fahd Al-Fatish
(fahd.alfatish@andrena.de)
ist Geschäftsfeld-Leiter Consulting bei der andrena objects ag und von Scrum.org zertifizierter Professional Scrum Trainer. Seit über 15 Jahren begleitet und coacht er Teams, Management und Organisationen bei der Einführung von agilen Methoden und der Skalierung von Agilität.