Evaluierung einer Experience Factory bei der COMPRA GmbH

Lennard Brunke

26. November 2022

Inhaltsverzeichnis

Zusammenfassung

In der vorliegenden Projektarbeit soll evaluiert werden, ob die DISER Methodologie geeignet ist, um bei der COMPRA GmbH eine neue Experience Factory zu entwickeln. Dieses System soll im Bereich der Softwareentwicklung des ERP Systems eEvolution und anderer Projekte der COMPRA GmbH Entscheidungsunterstützung bieten. (Tautz, 2001) Es wird die aktuelle Situation des Experience-Managements bei der COMPRA GmbH analysiert und im weiteren behandelt werden, wie die DISER Methodologie grundsätzlich aufgebaut ist und wie sie auf die Anforderungen der COMPRA GmbH sowie eine moderne agile Arbeitsweise angepasst werden kann. Das Ziel dieser und weiterführender Arbeit ist es, eine neue Experience Base bei der COMPRA GmbH aufzubauen.

Einleitung

Warum ist Erfahrungsmanagement für Unternehmen, insbesondere im Bereich der Softwareentwicklung, so wichtig? Reicht es aus, dass Wissen und Erfahrung im persönlichen Gespräch von Mitarbeiter zu Mitarbeiter weitergegeben oder sie sporadisch und in unterschiedlichsten Formen in einem Netzwerk, in einer Quellcodeverwaltung oder einem DMS abgelegt werden?

Wie kann sichergestellt werden, dass das Wissen des eines langjährigen Mitarbeiters nicht gänzlich verloren geht, wenn dieser das Unternehmen verlässt? Diese einleitenden Fragen geben einen guten ersten Eindruck, warum Erfahrungsmanagement im Bereich Softwareentwicklung eine zentrale Rolle für die Produktivität der Mitarbeiter und die Qualität der entwickelten Produkte spielt.

(Tautz, 2001) beschreibt in der Einleitung, dass der größte Vorteil im Wettbewerb von Unternehmen in der Softwareentwicklung darin besteht, bessere Software mit mehr Features schneller und mit geringeren Kosten entwickeln zu können. Dieses Know-How kann durch die Konkurrenz nicht imitiert werden, da es nicht von dem Softwareprodukt ableitbar ist. Daraus lässt sich annehmen, dass Experience Management im Software Engineering ein entscheidender Vorteil ist, um sich von der Konkurrenz abzusetzen.

Experience Factory

> In diesem Kapitel soll das Konzept von QIP und der Experience Factory erläutert werden.

Eine Experience Factory (Althoff und Birk, 1997) (Tautz, 2001) beschreibt eine Einheit in einem Softwareunternehmen, welche kontinuierliche und inkrementelle Verbesserung der Softwareentwicklung im Unternehmen erzeugen soll. Dies wird erreicht durch die Verarbeitung und Analyse aller Arten von Wissen und Erfahrung, welche im Kontext der Softwareentwicklung relevant sind. Sie dient als ein Repository (Experience Base) für dieses Wissen und entspricht einer Fallbasis im Sinne von Case-Based-Reasoning. Weiterhin ist die Experience Factory zuständig für die Administration der Experience Base.

In Kapitel 1.1 beschreibt Tautz das Konzept der Experience Factory durch einen Vergleich des CBR Zyklus mit dem Paradigma QIP.

Quality Improvement Paradigm (Basili, Caldiera und Rombach, 2002) ist ein früher Ansatz für iterative Qualitätsverbesserung von Softwareprojekten. QIP zielt darauf ab, schon in der Startphase eines Softwareprojektes festzulegen, welche Erfahrungen daraus gewonnen werden können und wie dies konkret erreicht werden kann. QIP ist als eine Feedback Loop aufgebaut, welche durch die selbst auferlegten Ziele gesteuert wird. Im letzten Schritt von QIP wird die gewonnenen Erfahrungen in eine wieder verwendbare Form gebracht und aufbewahrt. Im Kontrast dazu ist der CBR Zyklus mehr technischer Natur, aber grundsätzlich zielen beide Ansätze auf das Gleiche ab: Ein Experience Model welches im Kontext der Softwareentwicklung wiederverwendet werden kann.

(Basili, Caldiera und Rombach, 2002) definiert die Experience Factory folgendermaßen:

> The experience factory is a logical and/or physical organization that supports project developments by analyzing and synthesizing all kinds of experience, acting as a repository for such experience, and supplying that experience to various projects on demand. It packages experience by building informal, formal or schematized, and productized models and measures of various software processes, products, and other forms of knowledge via people, documents, and automated support.

Die genannten Konzepte wurden in einer Zeit entwickelt, in welcher das häufigste Vorgehen in Softwareprojekten einer Form des Wasserfallmodells entsprach. Heutzutage, und auch bei der COMPRA GmbH, wird ein agiler Entwicklungsprozess bevorzugt. Der Fakt, dass CBR und QIP schon damals einer iterativen Funktionsweise zugrunde lagen zeigt, dass ein agiler, iterativer Entwicklungsansatz diese Arten von Prozessen noch viel besser einsetzen

und einfacher Wert daraus gewinnen kann. Im Vergleich zum Gebrauch in einem Wasserfallmodell werden die Erfahrungsinkremente bei einem agilen Prozess kleiner, aber einfacher zu verarbeiten sein.

Dies lässt grundsätzlich vermuten, dass der Aufbau und letztendlich auch der laufende Betrieb einer Experience Factory sich gut in den aktuellen agilen Entwicklungsprozess der COMPRA GmbH integrieren lässt.

Experience Repositories und Experience Management

> In diesem Kapitel soll ein Überblick über die verschiedenen Ansätze, welche im Erfahrungsmanagement in der Softwareentwicklung genutzt werden, gegeben werden. Außerdem sollten hier die Technicken beschrieben werden, die bisher bei der COMPRA GmbH für das Thema Erfahrungsmanagement verwendet werden.

Bei der COMPRA GmbH werden verschiedene Softwaresysteme geführt, welche neben ihrer hauptsächlichen Funktion auch als Repository für Wissens- und Erfahrungsartefakte verwendet werden. Dies geschieht nicht zwangsläufig mit der Intention des Erfahrungsmanagement, sondern ist häufig eine Konsequenz aus Kommunikation und Kollaboration. Nachfolgend werden alle Softwaresysteme genannt, welche in irgendeiner Form als Repository für Erfahrungsmanagement genutzt werden:

- MS Team Foundation Server:

  Source Control Interne sowie externe Kollaborationsplattform zur Planung, Kommunikation und Dokumentation der Softwareentwiclung in allen Teams und Projekten Projekt- oder Teambezogene Wikis

- ELO Document Management System

  Interne Dokumente (Personalmanagement, Gehaltsabrechnungen etc.) Dokumentation von internen Prozessen und Vorgängen (Sprint Retrospektiven, Protokolle)

- Windows File-Server:

  Ablage von verschiedensten Dateien und Dokumenten Installationsdateien Projektbezogene Dateien und Dokumente

- ERP System eEvolution und MS SQL Server:

Erstellung von Rechnungen Dokumentation von Arbeitszeiten Verwaltung von Mitarbeitern und Kunden CRM

- eEvolution Wiki

  Offizielles, zentrales Wiki für das ERP System eEvolution

- MS Teams:

  Interne und externe Kommunikation und Kollaboration (Chats, Meetings).

- MS Sharepoint:

  Teamübergreifende Kalender, Dokumentation, Verschiedenes.

- Yammer:

  Internes soziales Netzwerk

- Email und IP-Telefon

Vergleichen wir diese Systeme mit den von Tautz aufgeführten vorhandenen Ansätzen für Erfahrungs-Repositories wird deutlich, dass eine ganze Reihe bei der COMPRA GmbH im Einsatz sind:

- Controlled / Uncontrolled Keyword System [Tautz001 5.1.4]
- Full Text Search auf vorhandene Repositories [Tautz001 5.1.5]
- Hypertext [Tautz001 5.2]
- Relational and Object-Oriented DBMS [Tautz001 5.3.1 5.3.2]

Alle vorhandenen Repositories verfügen über die Möglichkeit über Full Text Search durchsucht zu werden. Der Team Foundation Server verwendet ein unkontrolliertes Keyword System, wobei jegliche Art von Workitem mit beliebigen Keywords versehen werden kann. Das eEvolution Wiki ist eine Website, verwendet also Hypertext. eEvolution ist ein ERP System, welches mit einer relationalen Datenbank betrieben wird.

Auffallend ist hier, dass nur eher simple Ansätze verwendet werden, welche ohne weiteren größeren Aufwand innerhalb der genannten Softwaresysteme betrieben werden können. Es gibt kein System, welches logikbasiert, wissensbasiert oder mit irgendeiner Form von KI arbeitet.

Erfahrungsmanagement Ist-Situation

Zuvor wurden Erfahrungs-Repositories bei der COMPRA GmbH genannt. In diesem Kapitel soll im Detail behandelt werden, wie die Repositories im Prozess der Softwareentwicklung eingebunden sind, welche Vor- und Nachteile dies beinhaltet und welche Maßnahmen nötig sind, um die Softwareentwicklung durch gezieltes Erfahrungsmanagement zu verbessern.

Das am häufigsten und am erfolgreichsten verwendete System bei der COMPRA GmbH ist der MS Team Foundation Server (TFS). Er wird als Source Control, Product Backlog sowie Kommunikations- und Kollaborationsplattform (intern und extern) verwendet. Die COMPRA GmbH arbeitet nach einem agilen Manifest mit Scrum (Schwaber und Sutherland, o. J.). Im TFS wird das Product Backlog zu jedem Team bzw. Kunden oder Projekt in Form von Features, User Stories und Issues gepflegt. An diesem Product Backlog sind alle relevanten Stakeholder beteiligt.

Nehmen wir nun an, ein Team steht vor einer neuen Softwareanforderung und möchte frühere Entwicklungen und Erfahrungen nutzen, welche bereits für andere Projekte in diesem Bereich gemacht wurden, um den Entwicklungsprozess in den Aspekten Kosten, Geschwindigkeit und Qualität zu verbessern.

Das bisherige übliche Vorgehen ist hierbei, zunächst den TFS über eine Full Text Search, Keywords oder gefiltert nach Attributen des Product Backlogs zu durchsuchen. Dies kann in manchen Fällen schnell relevante Product Backlog Items finden, jedoch nur, wenn der Suchende bereits Kenntnis über die Existenz der relevanten Backlog Items hat. Ist keine Kenntnis darüber vorhanden, ist die Suche meist langwierig und ohne Erfolg. Aus diesem Grund wird vor dem Schritt des Durchsuchens des Backlogs im persönlichen Gespräch mit anderen Mitarbeitern versucht, bereits zu filtern ob ähnliche Anforderungen in der Vergangenheit bereits existierten, für welches Projekt bzw. Kunden diese entwickelt wurden und welche Mitarbeiter darin involviert waren. Mit diesen Informationen können die richtigen Kollegen angesprochen werden, um letztendlich die relevanten Backlog Items zu finden, sofern sie existieren. Es ist vorgesehen, dass Backlog Items mit sämtlichen relevanten Informationen verknüpft sind. Dies beinhaltet Code Changesets, Dokumentationen jeglicher Art sowie sämtliche Kommunikation zum Backlog Item im Bezug auf Requirement-Engineering. In der Praxis ist dies leider nicht immer der Fall und auch hier sind wichtige Informationen oft nur über die involvierten Mitarbeiter zu finden. Ein Problem dabei ist, dass wie oben beschrieben noch viele weitere Systeme als Experience-Repository verwendet werden und nicht alle Informationen im relevanten TFS Backlog

Item gesammelt werden. Gibt es beispielsweise eine Installationsanleitung mit “lessons learned” aus vorherigen Projekten, welche aber in einem Projektordner auf einem File Server abgelegt ist, so muss der Mitarbeiter erst auf diese Information hingewiesen werden.

Dies führt zu einer großen Belastung von gewissen Senior Mitarbeitern, welche den größten Erfahrungsschatz besitzen. Das Problem und die Auslastung wird noch viel größer, wenn der Mitarbeiter wiederholt aus verschiedenen Quellen nach den selben Informationen gefragt wird.

Um dieses Problem zu lösen, sollte ein einheitlicher Prozess für eine Experience Factory entwickelt werden, damit Artefakte in genau einem Repository zu finden sind und Mitarbeiter wissen, wie diese Artefakte dort abgelegt werden sollen. Weiterhin muss die Experience Factory in die Entwicklungsprozesse integriert werden, damit die Experience Base kontinuierlich wächst und verbessert wird. Wenn ein Mitarbeiter nach einer Information sucht muss klar sein wie und wo diese Information zu finden ist.

Agiler Entwicklungsprozess bei der COMPRA GmbH

In diesem Kapitel wird der agile Entwicklungsprozess bei der COMPRA GmbH aufgezeigt und weiterhin ermittelt, wie der Betrieb einer Experience Factory in diesen integriert werden kann.

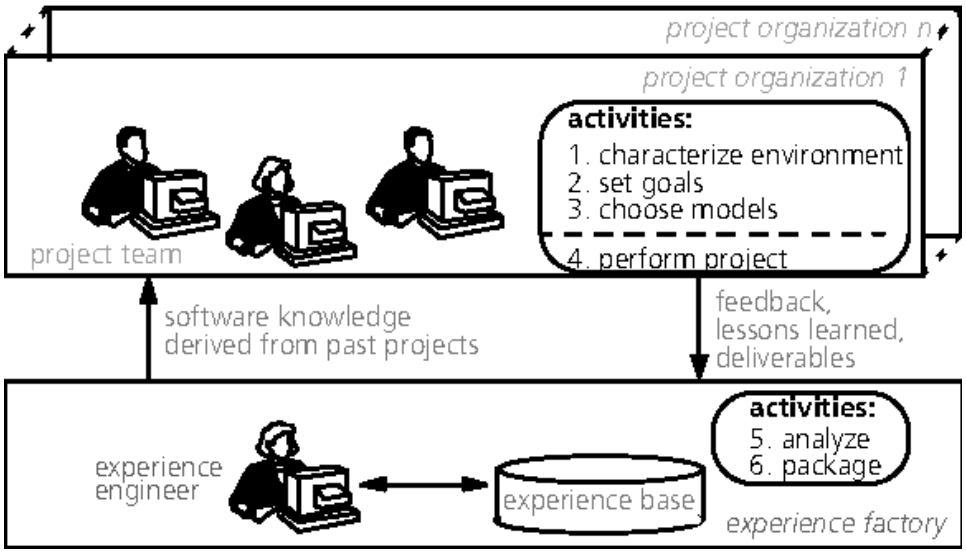
Die COMPRA GmbH setzt für die Komposition der Teams auf Full-Stack Teams. Das bedeutet, dass die einzelnen Teams zwar auf gewisse Bereiche spezialisiert sind, weil in diesen Bereichen einfach mehr Erfahrung vorhanden ist, grundsätzlich aber ein einzelnes Team alle Arten von Aufgaben oder Prozessen durchführen kann, welche in der Softwareentwicklung der COMPRA GmbH benötigt werden, also von der Akquirierung eines Projektes über Requirements-Engineering bis zum fertigen Softwareinkrement. Dabei arbeiten die Teams nach einem agilen Manifest mit Scrum (Schwaber und Sutherland, o. J.), wobei hier bei allen Teams gewisse Details unterschiedlich gehandhabt werden, der Kernprozess aber der Gleiche ist.

Als Beispiel betrachten wir das Allstars Team mit dem Projekt Rossmann Central Europe. Das Allstars Team entwickelt und betreut den Betrieb einer kundenspezifische Version des ERP Systems eEvolution 10 für Rossmann Ungarn und Rossmann Tschechien. Das Team besteht aus einem Scrum Master aus einem anderen Team bei der COMPRA GmbH, einem Product Owner von Rossmann Ungarn, einem Product Owner von Rossmann Tschechien und 5 Entwicklern sowie einem Auszubildenden. Das Product Backlog im Team Foundation Server ist eine Liste von Features, User Stories und Issues und stellt die zentrale Quelle an Arbeit für das Team dar. Ein Feature ist

hier gleichzusetzen mit einem Product Goal und beschreibt den Zielzustand eines Teiles des Produktes. Um diesen Zielzustand zu erreichen werden User Stories unterhalb eines Features definiert, welche umgesetzt werden müssen um das Ziel des Features zu erreichen. Dabei sollten User Stories im Umfang idealerweise möglichst klein sein, damit sie innerhalb eines Sprints abgeschlossen werden können und der Aufwand für dieses Workitem besser abschätzbar ist. Issues sind für ungeplante Arbeit, also Probleme welche während eines Sprints auftauchen und schnell gelöst werden müssen. Dies kann alle Arten von Problemen beinhalten welche durch das Scrum Team gelöst werden. (Bugs in laufender Software, Support für User etc.) Sprints werden grundsätzlich für eine Dauer von zwei Wochen geplant. Zum Ende eines Sprints werden sowohl ein Sprint Review, eine Retrospektive als auch ein Planning für den nächsten Sprint durchgeführt.

Ausgehend von der oben beschriebenen Situation kann festgestellt werden, wie eine Experience Factory in den Prozess integriert werden kann. Der erste Schritt zur Integration sollte sein, die Definition of Done eines Features, einer User Story oder eines Issues darum zu erweitern, das Wissen und die Efahrung aus diesem Workitem in den Prozess der Experience Factory einfließen zu lassen. Weiterhin gilt es zu entscheiden, welche Mitarbeiter aktiv für den Aufbau und den Betrieb der Experience Factory zuständig sind und wie sich diese als Einheit oder Team in den Rest des Unternehmens einbindet.

In (Tautz, 2001) und (Althoff und Birk, 1997) wird die Experience Factory als eine eigene Organisationsstruktur mit dedizierten Experience Engineers beschrieben (siehe Abbildung 1).



Für die COMPRA GmbH als mittelständiges, agiles Unternehmen mit Full-Stack Teams ist es unangebracht, Experience Engineers in einem eigenen Team zu beschäftigen. Stattdessen sollte es pro Team einige wenige Mitarbeiter geben, welche als zusätzliche Rolle aktiv für den Prozess der Experience Factory verantwortlich sind. Im Sinne von Agile würde dies im Unternehmen eher als eine Guild oder ein Chapter beschrieben werden. Eine Gilde ist eine Gruppe von Mitarbeitern aus verschiedenen Teams, welche gemeinsam an einem gewissen Thema oder einer Kompetenz arbeiten und sich regelmäßig dazu austauschen. Die Konzepte von Guilds, Chapters und Tribes wurde erstmalig von Spotify eingesetzt, um Agile in großen Unternehmen besser skalierbar zu machen. (Wardt, 2022) Ziel einer Gilde ist es, Erfahrung auszutauschen und das Unternehmen als ganzes zu verbessern. Sie sind also auch in kleineren Unternehmen sinnvoll und werden bereits von der COMPRA GmbH eingesetzt. Beispielsweise gibt es eine Consultant-Gilde und eine Entwickler-Gilde. Dies Ziele einer Gilde decken sich mit den Zielen der Experience Factory.

Die einzelnen Mitglieder der Gilde sind also dafür verantwortlich die Erfahrung aus der Arbeit des eigenen Teams in die Experience Factory zu bringen, um dann gemeinsam mit der Gilde diese Artefakte zu analysieren und zu verpacken, damit das Wissen im Unternehmen wieder verwendbar ist.

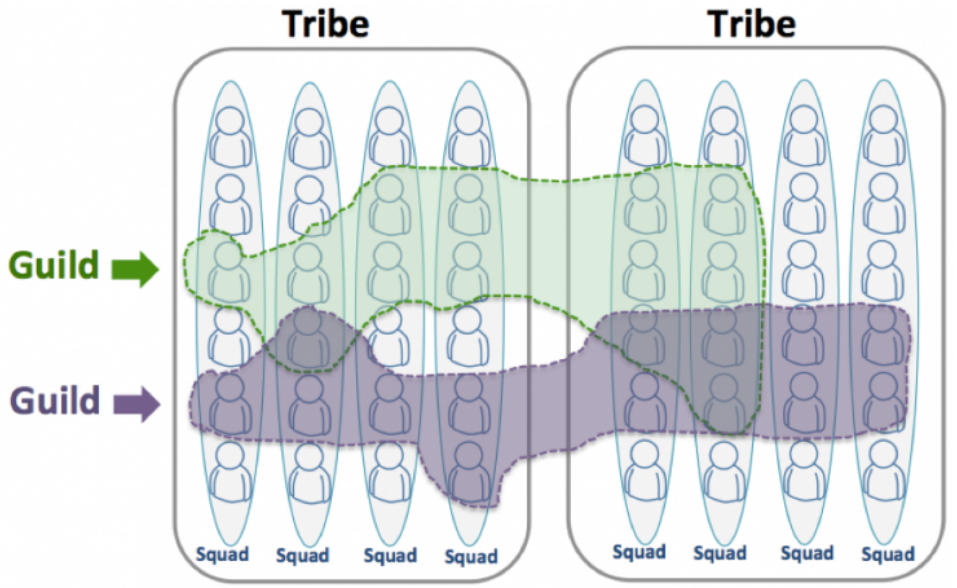


Abbildung 1: Agile und Scrum bei Spotify

DISER

In der Einleitung der Dissertation (Tautz, 2001) sagt Tautz, dass DISER auf dem Prnzip “learning from examples” basiert.

Tasks für den Aufbau einer Experience Base

In (Tautz, 2001, Kap.4) entwickelt Tautz ein allgemein gültiges Task Framework für den Aufbau und Betrieb einer Experience Base. Unternehmen können sich anhand dieser Task Hierarchie abhängig von ihren spezifischen Anforderungen selektieren, welche Aufgaben von der Experience Base ausgeführt werden sollen. Dabei definiert Tautz für jede Aufgabe das dadurch erreichte Ziel, Ein- und Ausgabe, die Anforderungen um die Tätigkeit ausführen zu können, eine Dekomposition in Unteraufgaben sowie eine allgemeine Beschreibung zur Ausführung der Aufgabe. In diesem Kapitel werden die Tasks selektiert, welche für den Betrieb einer Experience Base bei der COMPRA benötigt werden. Anhand dieser Tasks lassen sich die Anforderungen an das System festlegen. Hierbei werde ich nur eine kurze Zusammenfassung der jeweiligen Tasks sowie ihre Anforderungen und Dekomposition nennen, wobei die Dekomposition nur tatsächlich selektierte Aufgaben enthält. Weitere

Details sind (Tautz, 2001, Kap.4) zu entnehmen.
In Abbildung 3 ist die Task-Hierarchie zu sehen.

- **T2 reuse**
  Ziel: Die Wiederverwendung von Wissen und Erfahrung ist das zentrale Konzept der Experience Base und soll Qualität und Geschwindigkeit bei geringerem Aufwand erzielen.
  Anforderungen: R18 various characterizations of one artifact
  Dekomposition: T3 retrieve, T22 utilize

- **T3 retrieve**
  Ziel: Das Filtern der Experience Base nach relevanten Artefakten für die Problemstellung bzw. Suche.
  Anforderungen: keine
  Dekomposition: T4 specify, T8 identify, T11 evaluate, T19 select

- **T4 specify**
  Ziel: Formulierung eines Query für die Suche in der Experience Base
  Anforderungen: keine
  Dekomposition: T5 collect descriptors, 6 interpret problem, T7 infer descriptors

- **T5 collect descriptors**
  Ziel: Formulierung des initalen Query durch Beschreibung von Deskriptoren (Attributes, Keywords, Formulas)
  Anforderungen: R9 separation of characterization and conceptual information, R35 tolerance of incomplete query information, R36 tolerance of uncertain query information, R37 tolerance of imprecise query information
  Dekomposition: -

- **T6 interpret problem**
  Ziel: Das Query wird auf Fehler und Probleme geprüft.
  Anforderungen: R25 integrity constraints
  Dekomposition: -

- **T7 infer descriptors**
  Ziel: Das überprüfte Query wird durch das System vervollständigt.
  Anforderungen: R25 integrity constraints
  Dekomposition: -

- **T8 identify**
  Ziel: Auswahl potentiell relevanter Artefakte.

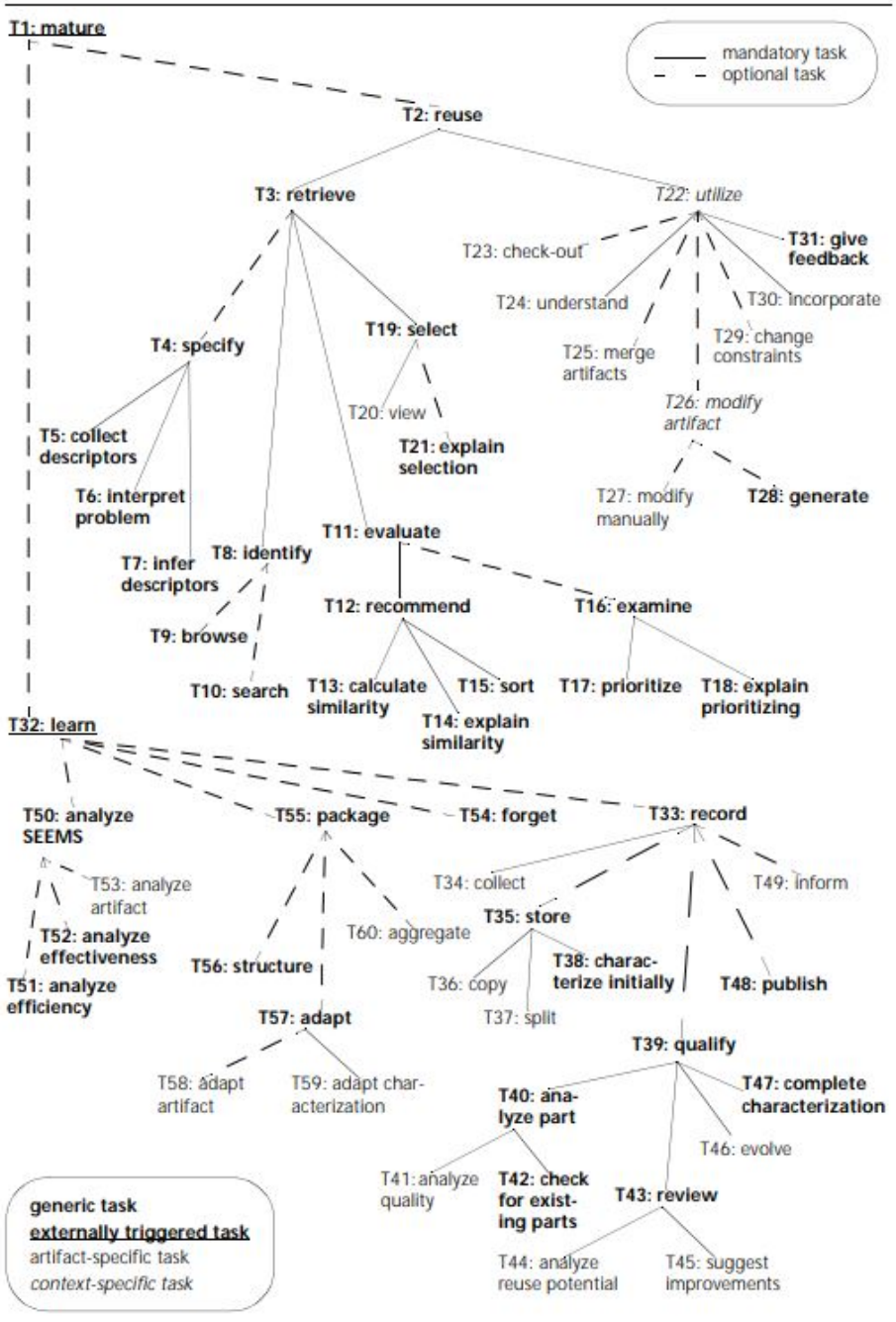


Abbildung 2: Task-Hierarchie

Anforderungen: keine
Dekomposition: T9 browse, T10 search

- **T9 browse**
  Ziel: Auswahl potenziell relevanter Artefakte durch manuelle Navigation (browsing).
  Anforderungen: R6 network access, R32 browsing
  Dekomposition: -

- **T10 search**
  Ziel: Auswahl potenziell relevanter Artefakte durch Suchalgorithmus (Ausschlusskriterien).
  Anforderungen: R6 network access, R7 storage of various kinds of artifacts, R27 artifacts' status
  Dekomposition: -

- **T11 evaluate**
  Ziel: Ausführung von T12 recommend.
  Anforderungen: keine
  Dekomposition: T12 recommend

- **T12 recommend**
  Ziel: Die gefundenen Artefakte werden sortiert nach ihrer Charakterisierung, um die hilfreichsten Ergebnisse zuerst anzuzeigen
  Anforderungen: keine
  Dekomposition: T13 calculate similarity, T14 explain similarity, T15 sort

- **T13 calculate similarity**
  Ziel: Berechne Ähnlichkeit zwischen finalem Query und einem Artefakt.
  Anforderungen: R9 separation of characterization and conceptual information, R20 tolerance of incomplete information, R21 tolerance of uncertain information, R22 tolerance of imprecise information, R23 transparency of duplicated information, R24 tolerance of inconsistent information, R33 textual search, R35 tolerance of incomplete query information, R36 tolerance of uncertain query information, R37 tolerance of imprecise query information, R39 context-sensitive retrieval
  Dekomposition: -

- **T14 explain similarity**
  Ziel: Erklärung der Sortierung durch Erklärung der Ähnlichkeit jedes

potenziellen Artefakts.
Anforderungen: R34 similarity-based retrieval
Dekomposition: -

- **T15 sort**
  Ziel: Sortiere potenzielle Artefakte absteigend nach ihrer Ähnlichkeit zum Query.
  Anforderungen: R34 similarity-based retrieval
  Dekomposition: -

- **T19 select**
  Ziel: Benutzer selektiert die am besten passenden Artefakte.
  Anforderungen: keine
  Dekomposition: T20 view

- **T20 view**
  Ziel: Detaillierte Betrachtung der potenziellen Artefakte durch den User und finale Selektion.
  Anforderungen: R3 tool integration, R5 access rights
  Dekomposition: -

- **T22 utilize**
  Ziel: Verwendung der selektierten Artefakte und daraus einen Mehrwehrt zu gewinnen.
  Anforderungen: keine
  Dekomposition: T23 check-out, T24 understand, T26 modify artifact, T27 modify manually, T30 incorporate, T31 give feedback

- **T23 check-out**
  Ziel: Auschecken des Artefakts durch Erstellung einer lokalen Kopie zur Bearbeitung.
  Anforderungen: R40 check-out of artifacts
  Dekomposition: -

- **T24 understand**
  Ziel: Herausfinden, wie die selektierten Artefakte am besten verwendet werden können.
  Anforderungen: R3 tool integration, R5 access rights, R6 network access, R29 configurations, R32 browsing, R41 interface information, R42 application history
  Dekomposition: -

- **T26 modify artifact**
  Ziel: Unterschied zwischen dem verwendeten Artefakt und der tatsächlich benötigten Erfahrung wieder in das Artefakt einfließen lassen
  Anforderungen: R3 tool integration
  Dekomposition: T27 mondify manually

- **T27 modify manually**
  Ziel: Editiere ein Artefakt um das tatsächlich benötigte Wissen.
  Anforderungen: R3 tool integration
  Dekomposition: -

- **T30 incorporate**
  Ziel: Das modifizierte Artefakt wird wieder in das System integriert.
  Anforderungen: R3 tool integration
  Dekomposition: -

- **T31 give feedback**
  Ziel: Feedback für die Verbesserung der Entscheidungen des Systems.
  Anforderungen: R42 application history
  Dekomposition:

- **T32 Learn**
  Ziel: Entscheidungsunterstützung durch das System effizienter und effektiver machen.
  Anforderungen: R2 support for incremental, continuous learning, R14 maintenance of experience packages
  Dekomposition: T33 record, T54 forget, T55 package

- **T33 record**
  Ziel: Einfügen neuer Artefakte in die Experience Base.
  Anforderungen: R7 storage of various kinds of artifacts, R26 accommodation of growing collection
  Dekomposition: T34 collect, T35 store

- **T34 collect**
  Ziel: Neue oder verbesserte Artefakte werden gesammelt.
  Anforderungen: keine
  Dekomposition: -

- **T35 store**
  Ziel: Neue Artefakte werden in der Experience Base zur weiteren Qualifikation gespeichert.

Anforderungen: keine
Dekomposition: T36 copy, T37 split, T38 characterize initially

- **T36 copy**
  Ziel: Neue Artefakte für lokale Modifikation verfügbar machen.
  Anforderungen: R3 tool integration, R5 access rights
  Dekomposition: -

- **T37 split**
  Ziel: Identifizierung der wiedervenwendbaren Teile eines Artefakts für die Charakterisierung.
  Anforderungen: R3 tool integration
  Dekomposition: -

- **T38 characterize initially**
  Ziel: Charakterisierung (manuell oder automatisiert) und Speicherung der wiederverwendbaren Teile eines Artefakts.
  Anforderungen: R3 tool integration, R5 access rights, R7 storage of various kinds of artifacts, R18 various characterizations of one artifact, R19 tolerance of different levels of abstraction, R20 tolerance of incomplete information, R21 tolerance of uncertain information, R22 tolerance of imprecise information, R24 tolerance of inconsistent information, R27 artifacts' status, R38 context information
  Dekomposition: -

- **T54 forget**
  Ziel: Entfernen von obsoleten Artefakten aus der Experience Base.
  Anforderungen: R27 artifacts' status
  Dekomposition: -

Repräsentation von Software Engineering Erfahrungen

> In diesem Kapitel soll (Tautz, 2001) Kapitel 6 zusammgengefasst werden.

Eine der elementarsten Fragen für den Aufbau einer neuen Experience Base ist, nach welchem Schema das Wissen und die Erfahrung strukturiert sein soll. Dies ist wichtig, um einerseits das Erfassen von Artefakten einfach und intuitiv zu halten und sie in der alltäglichen Kommunikation verwenden zu können, aber auch um die spezifischen Anforderungen des Unternehmens sowie der Softwareentwicklung abzudecken.

REFSENO

REFSENO (representation formalism for software engineering ontologies) (Tautz und Gresse von Wangenheim, 1998) beschreibt wie die Informationen in einer Experience Base formuliert und strukturiert werden können, um optimal für eine Experience Base im Bereich der Softwareentwicklung genutzt werden zu können. (Tautz, 2001, Kap.6) Es ist eine Kombination aus Ideen aus den Bereichen Datenbanken, fallbasiertes Schließen und wissensbasierten Systemen. Es ist ein objektorientiertes Modell und hat Ähnlichkeiten zu UML.

REFSENO's zentrales Element wird durch "concepts" definiert, welche Software Engineering Artefakte modellieren. Concepts setzen sich durch Attribute zusammen, welches entweder Informationen für die Speicherung und das Retrieval enthalten oder Relationen zu anderen Artefakten beschreiben (terminal and non terminal attributes). Weiterhin enthält REFSENO Formlen für die Definition der Ähnlichkeit (similarity), automatische Berechnung von Attributen (value inferences) und weitere Bedingungen, welche immer als true evaluiert werden müssen (assertations).

GOODSEE (goal-oriented ontology development for software engineering experience)

Technische Infrastruktur einer Experience Base

In (Tautz, 2001, Kap.3) wird beschrieben, welche Anforderungen an die technische Infrastruktur einer Experience Base gestellt werden. In Abbildung 4 werden alle Anforderungen und ihre Abhängigkeiten aufgelistet.

In der Zusammenfassung des Kapitels benennt Tautz R12, R38 und R39 sowie all ihre Abhängigkeiten als die elementarsten für den Aufbau einer Experience Base. Daraus ergibt sich die folgende Liste an Anforderungen:

- R7 storage of various kinds of artifacts
- R10 rationalized conceptual information
- R12 accomodation of new artifact kinds
- R11 modularity of conceptual information
- R17 artifact recording
- R27 artifacts' status
- R28 versioning
- R29 configuration
- R34 similarity-based retrieval
- R38 context information

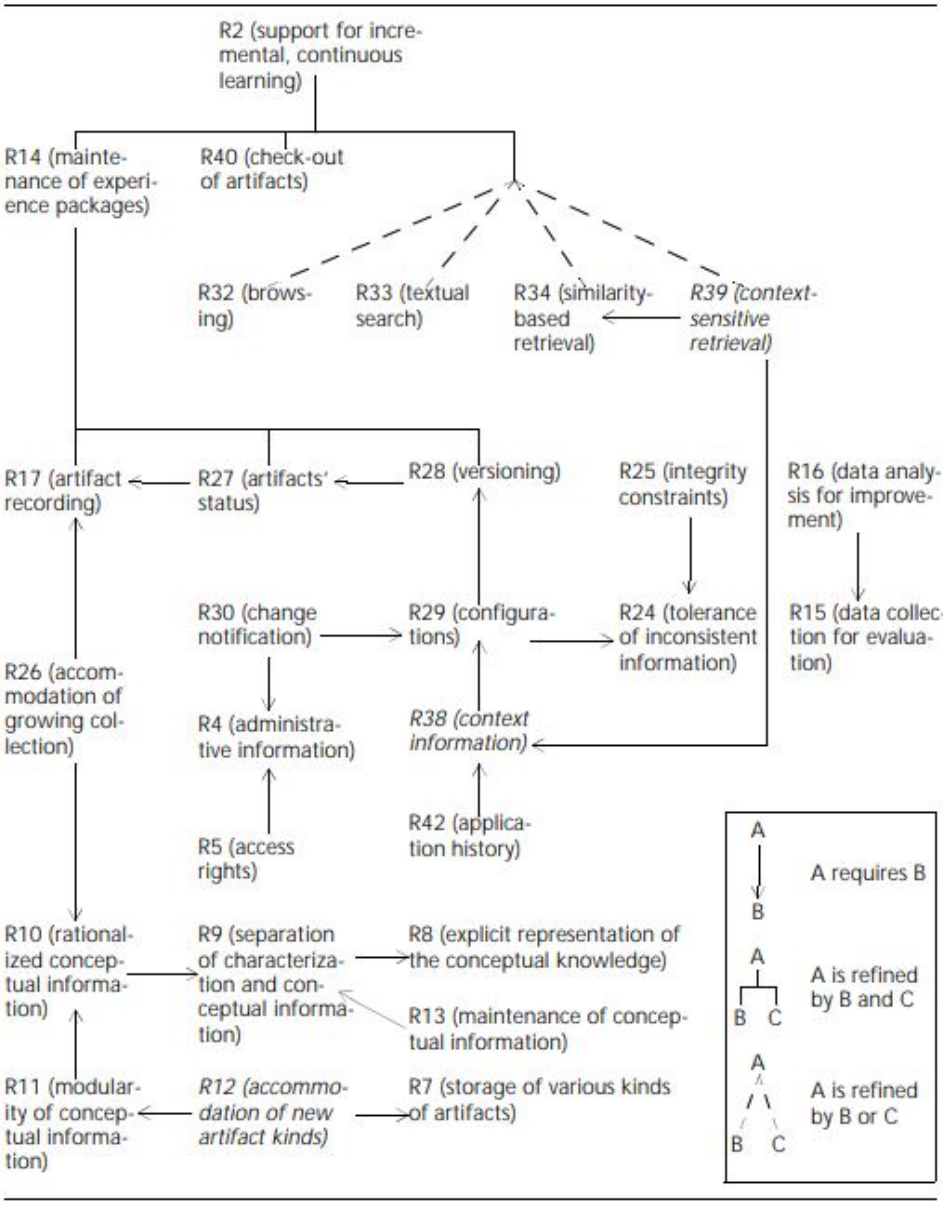


Abbildung 3: Anforderungen technische Infrastruktur

- R39 context sensitive retrieval

Diese Anforderungen bilden nach Tautz das Minimal Viable Product für eine Experience Base.
R12 stellt sicher, dass neue Arten von Artefakten angelegt werden können. Da eine Experience Base in einem Unternehmen über einen längeren Zeitraum wächst, wird auch die Art und Struktur der Erfahrungs-Artefakte sich im Laufe der Zeit ändern.
R38 bestimmt, dass überhaupt Artefakte in der Experience Base angelegt und gespeichert werden können. R39 besagt, dass nur Artefakte beim Retrieval zurückgegeben werden, welche für das Projekt bzw. den Kontext relevant sind. Tautz sagt, dass R39 vor seiner Dissertation von keiner technischen Infrastruktur unterstützt wurde und die Dissertation hier zum state-of-the-art beiträgt.

Für die COMPRA GmbH soll dieser kompakte Grundsatz an Funktionalität für die erste Implementierung einer Experience Base genügen.
Es ist vorteilhaft, solch ein neues System nicht mit zu vielen Features im ersten Prototypen zu entwickeln, um früh auf Feedback der Mitarbeiter eingehen zu können und die weitere Entwicklung diesem Feedback anpassen zu können, damit das System letztendlich für die COMPRA GmbH den größten Nutzen hat.

In (Tautz, 2001, Kap.7) wird eine Implementierung beschrieben mit welcher die Anforderungen an die technische Infrastruktur umgesetzt werden können.
Um die Anforderungen zu erfüllen verwendet Tautz eine Client-Server Architektur, welche in Abbildung 4 zu sehen ist.

Die Architektur ist in drei Ebenen unterteilt:
* 1. Anwendungsebene * 2. Zugriffsebene * 3. Datenbankebene

Auf der Anwendungseben befinden sich alle Arten von Software, mit welchen der User interagiert. Diese werden unterteilt in Artefakt Tools und Experience-Base Tools.
Artefakt Tools sind für die Entwicklung und Wartung von spezifischen Arten von Artefakten zuständig (Text/Dokument Editor, Prozess Modellierung etc.).
Experience-Base Tools werden für die Wartung der Experience Base verwendet. Sie sind unabhängig von der Art der Artefakte (General Purpose Browser für die Manipulation von Einträgen in der Experiece Base, Analyse Tools).

Die Zugriffebene ist für die Synchronisation zwischen Anwendungs- und Datenbankebene verantwortlich. Sie steuert Zugriffsrechte, erstellt Statistiken

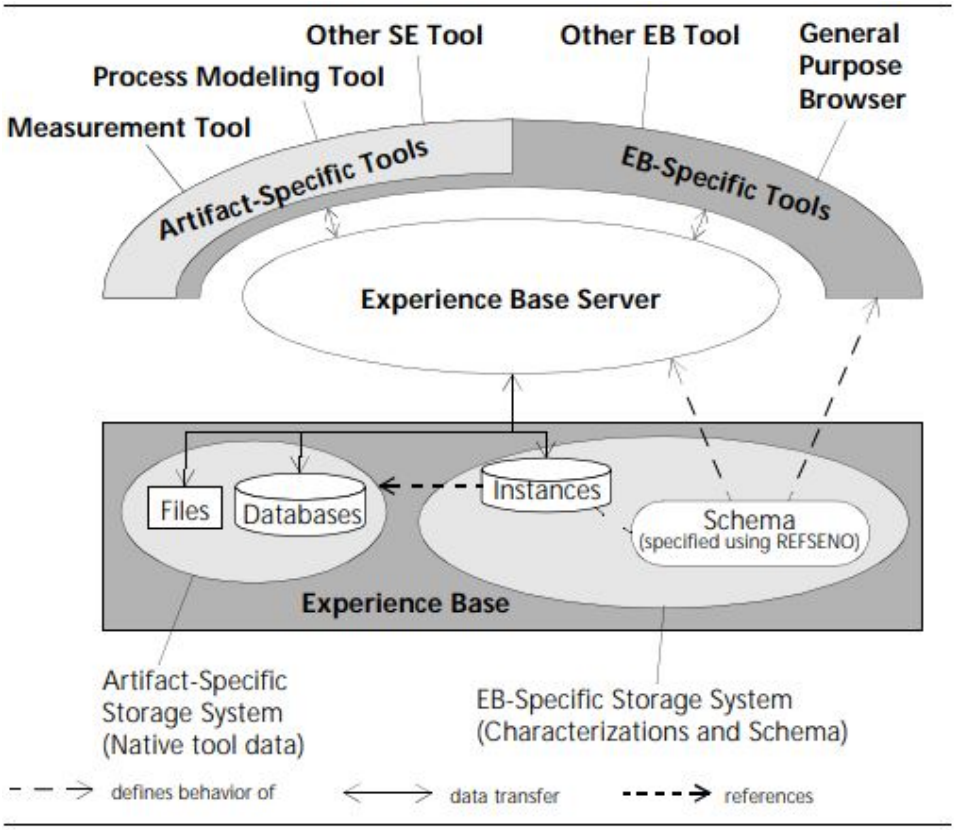


Abbildung 4: Architektur der technischen Infrastruktur

über Zugriffe und speichert Feedback für die Verbesserung der Infrastruktur.

Die Datenbankebene enthält die Datenbanksysteme für Artefakt-spezifische Daten (im Artefakt nativen Format) und Experience-Base-spezifische Daten (Charakterisierung und Schema (REFSENO)).

Technische Infrastruktur COMPRA GmbH

Grundsätzlich ist eine Client Server Architektur für die COMPRA eine sinnvolle Wahl. Die beschriebene technische Infrastruktur hat den Vorteil, dass sie an die spezifischen Anforderungen der COMPRA angepasst werden kann. Für die Anwendungsebene bedeutet dies, dass aktuell bereits eingesetzte Software für die Experience Base verwendet werden kann. Der wichtigste Teil in der Anwendungsebene für die COMPRA GmbH ist dabei der Team Foundation Server, welcher über ein Web-basiertes UI verwendet wird. So sollte es für die Workitems des Team Foundation Servers möglich sein, diese in die Experience Base zu importieren und daraus generierte Artefakt bearbeiten zu können.

DISER Task Framework

DISER Anforderungen

Wissensquellen

Fazit

Relevante Kapitel aus [Tau001]

T32 LEARN Decomposition: record (T33, page 107), analyze SEEMS (T50, page 116), forget (T54, page 119), package (T55, page 120)

Literaturverzeichnis

Althoff, K.-D. und Birk, A., 1997. The Experience Factory Approach: Realizing Learning From Experience In Software Development Organizations.
Basili, V., Caldiera, G. und Rombach, D., 2002. Experience Factory. In: *Encyclopedia of Software Engineering*. https://doi.org/10.1002/0471028959.sof110.
Schwaber, K. und Sutherland, J., o. J. *The 2020 scrum GUIDE. Scrum Guide | Scrum Guides*. Verfügbar unter: <https://scrumguides.org/scrum-guide.html>.

Tautz, C., 2001. *Customizing software engineering experience management systems to organizational needs.* phdthesis.
Tautz, C. und Gresse von Wangenheim, C., 1998. *REFSENO: A representation formalism for software engineering ontologies.*
Wardt, R. van der, 2022. *Das Spotify Modell: Agile und scrum für Große Organisationen. Agile Scrum Group.* Verfügbar unter: <https://agilescrumgroup.de/spotify-modell/>.