Das Teilsystem Lehre

2

Eine „Big-Bang“ Einführung großer soziotechnischer Softwaresysteme ist hoch riskant, insbesondere bei Neuentwicklungen [57]. Daher sollte stufenweise implementiert werden. Unter *Implementierung* verstehen wir den technischen *und* den organisatorischen Vorgang [20, 54, S 73]. Um die Möglichkeiten und Grenzen einer stufenweisen Einführung zu untersuchen, muss die Abhängigkeit zwischen Funktionen und Datenbasis betrachtet werden. In erster Näherung nehmen wir drei funktional sichtbare Bausteine eines CMS an – Lehre, Forschung und spezielle Dienste (insbes. Bibliothek) – und beginnen mit dem ersteren. In diesem Teilsystem liegt offenbar seit der Änderung der Hochschulgesetzgebung („BA/MA“) auch die höchste Dringlichkeit. Es wird sich zeigen, dass dieses Teilsystem u. a. wegen der Heterogenität seiner Benutzer und der Spitzenlasten bei Buchungen das komplexeste ist.

2.1 Funktionen, Daten und Integrationskern

Datenbasis und Funktionen

Tabelle 2.1 betrachtet die Funktionen eines CMS mit den korrespondierenden Daten. Die Grunddaten sind starke Entitätstypen, die von anderen nicht abhängig sein dürfen. Die Mengengerüste der Grunddaten sind relativ gering, im Gegensatz zu Vorgangsdaten. Die Masse der Daten entsteht in den „betrieblichen“ Vorgängen. Sie sind als schwache Entitätstypen von mindestens einem Grunddatentyp abhängig. Spitzenlasten durch neue Vorgangsdaten entstehen zu Beginn und zum Ende von Semestern (Belegen von Veranstaltungen und Buchen von Prüfungen). Die Server-Infrastruktur muss diese Transaktionen bewältigen, da sonst wichtige Arbeitsprozesse behindert werden.

© Springer Fachmedien Wiesbaden 2015

T. Spitta et al., *Campus-Management Systeme als Administrative Systeme,*
essentials, DOI 10.1007/978-3-658-11595-1_2

Tab. 2.1 Funktionen, originäre und einige abgeleitete Daten eines CMS

Funktion	***--- originäre Daten ---***		***abgeleitete Daten***
Verwaltung von ...	***Grunddaten***	***Vorgangsdaten***	(Beispiele)
.. Studierenden	Student	Rück-/ Anmeldung	Studienfortschritt
.. Lehrveranstaltungen	Lehrangebot	Teilnahme	Teilnehmer-Statistik
.. Räumen	Raum	Belegung	Kennzahlen Belegung
.. Studiengängen	Studiengang	Vorlesungsverzeichnis	Studierenden-Statistik
.. Prüfungen	Leistungs-Regelungen	Leistungs-Exemplar	Zeugnis
.. individuellen Stundenplänen	Stud+La+Stg+Raum	Belegung	Überschneidungsquote

Die abgeleiteten Daten wiederum sind überwiegend auf Vorgangsdaten aufgebaut, häufig in verdichteter Form. So baut etwa ein Zeugnis beim BA/MA-Lehrkonzept auf einer Vielzahl studienbegleitender Leistungen auf.

Die letzte Zeile von Tab. 2.1 zeigt ein wichtiges Beispiel, das eher *nicht* für zentrale Planer, Dozenten oder Fakultäten relevant ist, sondern für die Stakeholder, für die das Ganze stattfindet, die Studierenden. Bei hoher Wahlfreiheit ist es sehr schwierig und in jedem Einzelfall sogar kombinatorisch unmöglich, die Überschneidungsfreiheit jedes individuellen Stundenplans zu gewährleisten (s. hierzu [12, S. 486 ff.]). Bereits in einer mittelgroßen Universität wie Bielefeld werden seit der Einführung von BA/MA über 500 Studiengänge[1] studiert. Das Beispiel zeigt aber auch die Abhängigkeiten der Funktionen von Grunddaten. Wenn Teilsysteme für die ersten vier Grunddaten-Typen nicht existieren, kann die Funktion nicht angeboten werden, weil originäre Daten fehlen.

Abgeleitete Daten

Die Unterscheidung in originäre und abgeleitete Daten ist nicht neu [54], aber im Hochschulbereich wenig bekannt. Originäre Daten entstehen in der Umwelt eines Softwaresystems durch menschliche oder maschinelle Akteure. Maschinelle Akteure sind Maschinen, an denen Daten über Sensoren abgegriffen werden, etwa das Lesegerät für eine Mensa-Chipkarte. Die Software ist in die Realität (= Organisation) eingebettet und wirkt auf die Realität zurück, die sie modelliert. Wir werden diesen wichtigen Aspekt im folgenden Abschnitt vertiefen.

Die originären Daten bilden den Kern einer administrativen Datenbasis und sind auch Gegenstand eines entsprechenden Datenmodells, das wir in Kap.4 kurz

[1] 20 Kern- * 25 Nebenfächer = 500 Studiengänge. Diese haben noch je 3 Profile, was sich bei 4 Pflicht-SWS auf 6000 nötige Zeitfenster (ZF) ohne Überschneidungen aggregiert. Möglich sind aber pro Student nur 60 ZF/Woche bei 12 h pro Tag (8:00–20:00) (Beispiel von Maik Jablonski aus dem Vortrag zu [12]).

zeigen. Für die Korrektheit sind die Daten erzeugenden Akteure verantwortlich (einzelne oder Organisationseinheiten), die selbstverständlich so gut wie möglich durch Integritätsbedingungen vor Irrtümern geschützt und durch eine lückenlose Abbildung von Prozessen unterstützt werden müssen.

Abgeleitete Daten werden nicht durch Akteure der Organisation erzeugt, sondern durch *Programme*. Die Korrektheit des Programms verantwortet ein Programmierer, auch wenn gelegentlich der Benutzer Fachwissen zum Algorithmus oder zu Integritätsbedingungen beisteuern muss. Werden abgeleitete Daten ohne Rückgriff auf gespeicherte Daten erzeugt, verhalten sie sich wie eine Funktion $y = f(x)$, bei der das Ergebnis beliebig reproduzierbar ist. Basiert eine Funktion auf Zuständen (Gedächtnis aus der Datenbasis), so muss sichergestellt sein, dass sich die zu Grunde gelegten originären Daten nicht ändern[2]. Dies ist bei Grunddaten selten der Fall, bei Vorgangsdaten jedoch häufig, weil in Prozessen mit hoher Frequenz ständig neue Daten entstehen. Dies bedeutet, dass abgeleitete Daten nicht gespeichert werden sollten und heute in vielen Fällen dank der Hardwaretechnik auch nicht gespeichert werden müssen. Die simple betriebswirtschaftliche Formel

$$\text{Bestand}_t = \text{Bestand}_{t-1} + \text{Bewegung}_t$$

führt nur dann zu einem korrekten Bestand, wenn die oben skizzierte Operation in der *selben* Transaktion ausgeführt wird, wie die Buchung des (originären) Vorgangs *Bewegung*. Was bedeutet dies für ein CMS?

1. Wenn wir Software evolutionär einführen oder entwickeln, muss strikt darauf geachtet werden, dass korrekte und gut archivierte originäre Daten entstehen.
2. Voneinander existenzabhängige (originäre) Daten können nicht isoliert behandelt werden. Da fast alle relevanten Prozesse Vorgangsdaten erzeugen, können nur diejenigen unterstützt werden, deren Grunddaten mit geklärter Datenverantwortung existieren.
3. In einer Übergangsphase kann es sinnvoll sein, die dezentrale Erzeugung von Grunddaten zuzulassen, aber nur dann, wenn dies Einzelfälle sind und die Daten bei einer Weiterentwicklung kontrolliert integriert werden.
4. Die Ermittlung manchmal auch komplexer abgeleiteter Daten kann nachgeholt werden, wenn Zeit oder Budget knapp sind. Ein zeitliches Verschieben „infor-

[2] Selbstverständlich benötigt man dann Archive, die die zugehörigen originären Daten für lange Fristen Periodenbezogen festhalten, wie dies in Deutschland gem. HGB §§ 238 und 239 (4) für Handelsgeschäfte vorgeschrieben ist.

mativer“ Funktionen bei einer evolutionären Einführung kann die Integrität der zentralen Datenbasis nicht negativ beeinflussen.

Allerdings sollten sich Entscheider die Einführung eines Data Warehouse Systems [50] nicht einfach und nicht „billig“ vorstellen. Solche Systeme bieten nur dann einen Nutzen, wenn sie aus einheitlichen Datenquellen gespeist werden [67, S. 485 f.] und wenn die laufende Aktualisierung des Datenlagers aus originären Daten sorgfältig spezifiziert und sicher automatisiert wurde.

CMS als Eingebettetes System

Nach Lehmann [40] entwickeln sich embedded programs mit ihrer Umwelt. Dies zeigt Abb. 2.1 mit „R“ als Umwelt. Es bedarf für jede Version der Rückkopplung ($t_1 \ldots t_n$) einer entsprechenden Spezifikation. Der Begriff *Embedded Systems* wird in der Informatik nur für Software benutzt, die in technische Systeme eingebettet ist und sie überwacht und/oder steuert. Doch auch der Softwaretyp, den wir hier betrachten, kann als „eingebettet“ in eine Organisation gesehen werden [54].

Die Akteure der Umwelt des Softwaresystems sind die Benutzer, die originäre Daten erzeugen. Da die Organisation sich im Zeitablauf ändert, wird auch ein „fertiges“ Softwaresystem immerzu weiter gepflegt werden müssen. Wir nennen den hier betrachteten Softwaretyp *Administrative Systeme*, weil die überkommenen Begriffe entweder zu eng oder semantisch falsch und eher von Marketingüberlegungen geprägt sind (*Betriebliches* Informationssystem, Enterprise Resource *Planning* System ERP).[3] Die Systeme sind in Wirklichkeit Buchungssysteme für organisatorische Vorgänge, *keine* Planungssysteme. Aus ihnen werden Informa-

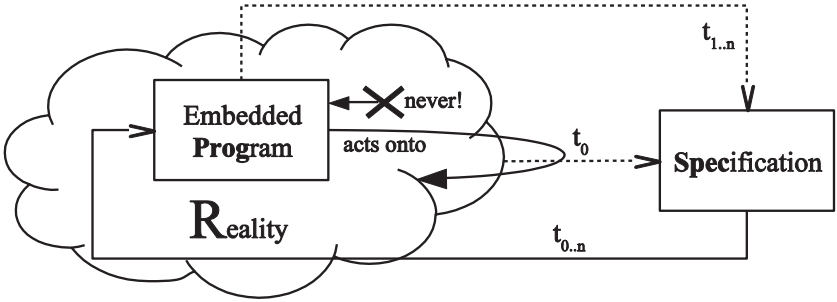

Abb. 2.1 Administrative Software als *embedded system*. (in Anlehnung an [54, S. 31])

[3] Mertens [44] benutzt den Begriff *Betriebswirtschaftlich-administrative Informationssysteme*.

tionen für Entscheidungen gewonnen. Selbstverständlich enthalten sie auch Planungsdaten, aber die Vorgangsdaten dominieren die operative Basis.

Warum haben wir so weit ausgeholt?

- Campus-Management Systeme sind schwergewichtige Administrative Systeme.
- Die wesentlichen Eigenschaften dieser Systeme müssen beachtet werden, damit eine evolutionäre Einführung erfolgreich sein kann, ganz besonders die Abhängigkeiten von Daten.
- Die Systeme müssen in der ersten Stufe möglichst schlank sein. „Schlank" kann bei verschiedenen Komponenten erhebliche Abweichungen haben.

Dies vermitteln Grechenig et al. durch Tab. 2.2, die CMS nach den Lehrbeispielen (No 1 und 2) in eine Reihe typischer großer Projekte stellt. Insbesondere die Zahl der Promotoren und der User machen ein CMS zumindest in einer großen Hochschule zu einem hoch komplexen System.

Durch die Einbettung in eine Organisation, in der die Studierenden *keine* „Kunden" in der Umwelt sind, sondern wesentliche Systemelemente, sollte man einen Plan haben, wie man die Software stufenweise in die Organisation implementiert. Die Teilsysteme sollten so einfach wie möglich sein. „Möglich" heißt, dass man die Abhängigkeiten der Datenbasis beachtet, zusätzliche Funktionalität aber aus dem Kernsystem heraus hält oder für spätere Ausbaustufen zurück stellt.

Ein CMS ist technisch (Lastspitzen bei Transaktionen)[4] und organisatorisch *schwergewichtig*. Die Komplexitätsreduktion der Einführung eines solchen Systems ist wünschenswert bis notwendig.

Tab. 2.2 Die Komplexität von Softwareprojekten [24, S. 91]

Projektkomplexität aus User- und Interessens-Dynamik						
Nr		**Promotors**	**Core-User**	**User**	**Visitors**	**User-Complexity**
1	Lehrbeispiel	1	2	10	200	0,01
2	Kleines Projekt	1	10	30	3.000	0,25
3	Mittleres Projekt	4	100	2.000	20.000	1
4	Große Hochschule	100	4.000	40.000	1 Mio	100
5	Tickets staatl. Bahn	15	8.000	4 Mio	20 Mio	200
6	ID-Paraguay	50	15.000	8 Mio	> 8 Mio	400
7	eHealth Germany	1.000	200.000	80 Mio	> 80 Mio	4.000

[4] Die Werte sind für industrielle Maßstäbe nicht der Rede wert. Man sollte aber beachten, dass Hochschul-Rechenzentren oft keine Erfahrungen mit dem Betrieb transaktionaler Datenbanken haben oder aus „Sparsamkeit" zu kleine Server betreiben.

2.2 Abhängigkeiten zwischen Administrativen Systemen

Der Datentyp *Dozent* ist ein wichtiges Beispiel für die Abgrenzung eines CMS von anderen administrativen Teilsystemen. Die Grunddaten derjenigen Personen, die ein Entgelt beziehen, müssen in dem System verwaltet werden, das den Datentyp „beschäftigte Person" verwaltet. Für eine Entgeltabrechnung und die entsprechenden Grunddaten wird man eine Standardsoftware einsetzen und die im CMS notwendigen Attribute von dort importieren. Die Datenverantwortung für Personaldaten muss bei der zuständigen Organisationseinheit liegen.

Eng mit einem **Personalsystem** verbunden ist ein Abbild der **Aufbauorganisation**. Sie ist jedoch in gängigen Standardsystemen wie etwa SAP ERP so stark mit dem Gesamtsystem verknüpft, insbes. der Kostenverantwortung [3, S. 54 f.], dass es nicht geraten erscheint, diese Strukturen für ein CMS zu nutzen, wenn aus dem ERP-System nur Finanzbuchhaltung und Personalsystem verwendet werden. Es sollte also eine Komponente *Organisation* geben, entweder im CMS oder eine unabhängige, die eng mit dem Personalsystem und der Kostenstellenstruktur gekoppelt ist. Es ist eigentlich selbstverständlich, dass *alle* Organisationsstrukturen in genau *einem* System abgebildet sind und nicht Verwaltung und Fachbereiche auseinander fallen.

Ein anderes Beispiel ist das **Gebäudemanagement** (*Facility Management*). Wenn ein FM-System betrieben wird, muss dieses bei den Räumen für Lehrveranstaltungen die entsprechenden Attribute pflegen (Medienausstattung, Plätze, Wartungsarbeiten etc.) und sie an das CMS exportieren. Existiert es nicht, kann es notwendig sein, im CMS eine Komponente *Raumverwaltung* zu haben, die aber keinesfalls „evolutionär" zu einem Gebäudemanagementsystem ausgebaut werden darf. Im Gegenteil, bei der Beschaffung eines solchen Systems muss diese Komponente des CMS „rückgebaut" werden.

Durch die Beispiele soll folgender Grundsatz für ein CMS vermittelt werden:

> Campus-Management Systeme müssen so schlank und modular wie möglich definiert werden, weil der transaktionale Kern des Teilsystems *Lehre* bereits schwergewichtig und ein Teilsystem *Forschung* (s. Kap. 3) ebenfalls notwendig ist.

Basisfunktionen und verteilte Grunddaten

In Tab. 2.1 auf Seite 6 sind nur anwendungsbezogene Funktionen genannt. Neben allgemein verfügbaren, technischen Basisfunktionen wie Dateitransfer (z. B. herunter laden von Lehrmaterial, herauf laden von Aufgabenlösungen) gibt es vor allem *eine* Basisfunktion, die in Administrativen Softwaresystemen von Wirtschaft

und Verwaltung keine auch nur annähernd so wichtige Rolle spielt wie in einem CMS: Die **Kommunikation** zwischen Lehrenden und Lernenden.

Die Kommunikationsfunktion sollte möglichst unabhängig von der Geschäftspolitik eines Herstellers sein. Jeder Lehrende muss alle Lernenden oder definierte Untergruppen seiner Veranstaltungen informieren und mit ihnen kommunizieren können. Die Partner dieser Kommunikation wechseln zumindest mit jedem Semester, in vielen Fällen auch laufend (z. B. angemeldete Teilnehmer eines Prüfungstermins). Die Studierenden sollten über das CMS selbst steuern können, über welchen Kanal die Nachrichten sie erreichen sollen, wenn dies auf Sicherheit überprüfte Kanäle sind.

Eng verbunden mit der im vorigen Abschnitt besprochenen Organisations-Komponente ist eine **Rechteverwaltung**, die für alle Administrativen Systeme selbstverständlich ist.

Eine weitere Basisfunktion ist z. T. technisch begründet. Sie bezieht die originären Daten aus der bereits im vorigen Abschnitt genannten Personalfunktion. Für fast alle Funktionen in Forschung und Lehre (F&L) wird im CMS eine Komponente ***Person-F&L*** benötigt, deren zentrale Grunddaten am besten aus dem Personalsystem repliziert werden. In den Teilsystemen Lehre, Forschung und Bibliothek wird die *Person-F&L* benötigt, die lehrt, prüft, forscht, publiziert etc. Diese Komponente darf natürlich nur diejenigen Personendaten aus dem zentralen System zeigen, die nicht dem Datenschutz unterliegen und nur jene replizieren, die für die Verarbeitung benötigt werden. Für den Datentyp *Person-F&L* sind dezentral schreibende Zugriffsrechte notwendig, und zwar disjunkt zu den zentralen. Z. B. wird so vermieden, dass eine Person nach der Habilitation im zentralen Personalsystem immer noch als Tutor geführt wird, die Rolle, mit der die Person vor langer Zeit im zentralen Personalsystem angelegt worden war. Hierdurch wird sicher gestellt, dass pro Attribut von *Person-F&L* nur *ein* System führend ist. Zusätzlich kann die Komponente *Person-F&L* den Studierenden dezentral gepflegte Kontaktinformationen zur Verfügung stellen, wie Sprechstunden und -orte.

Abgrenzung CMS – E-Learning

Allein die eben diskutierte Komplexität des Teilsystems Lehre macht es ratsam, eine Grenze zwischen den administrativen, standardisierbaren Vorgängen der Planung und Abwicklung von Lehre bis zur Prüfung zu ziehen und deren weiter gehender Unterstützung. Erstere gehören in das Teilsystem *Lehre* eines CMS, letztere können von ergänzenden E-Learning-Systemen flexibler geleistet werden. Die Grenze sollte so gezogen werden, dass E-Learning in autonomer Verantwortung des Dozenten optional die *interne* Gestaltung einer Lehrveranstaltung bis vor die Abschlussbewertung unterstützt – also das Lernen, nicht aber den Gesamtprozess der Lehre von der Anmeldung bis zur abschließenden Bewertung.

Die Grenzen zwischen zentralem CMS und eher dezentral genutzten E-Learning Systemen sind z. T. fließend, dürfen aber niemals zu einem Nebeneinander redundanter Grunddaten führen. Ein Beispiel ist die Einteilung großer Vorlesungen in Übungsgruppen mit der Abgabe von bewerteten Aufgaben. Dies könnte in einer naturwissenschaftlich und technisch geprägten Universität zum zentralen System gehören, in einer eher geisteswissenschaftlich geprägten aber in einem „angedockten" E-Learning-System abgewickelt werden.

Nach der Einführung eines zentralen CMS können auch generell benötigte Unterstützungsfunktionen mit entsprechenden Daten ergänzt werden, wenn sie Fächer übergreifend benötigt werden.

http://www.springer.com/978-3-658-11594-4

Campus-Management Systeme als Administrative Systeme
Basiswissen und Fallbeispiele zur Gestaltung und Einführung

Spitta, T.; Carolla, M.; Brune, H.; Grechenig, Th.; Strobl, S.; vom Brocke, J.

2015, IX, 37 S. 4 Abb., Softcover
ISBN: 978-3-658-11594-4