Technische Informationssysteme in der Energieversorgung : moderne Entwurfsverfahren

Autor(en): **Goudie, David**

Objekttyp: **Article**

Zeitschrift: **Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association Suisse des Electriciens, de l'Association des Entreprises électriques suisses**

Band (Jahr): **81 (1990)**

Heft 13

PDF erstellt am: **08.08.2024**

Persistenter Link: https://doi.org/10.5169/seals-903136

Technische Informationssysteme in der Energieversorgung

Moderne Entwurfsverfahren

David Goudie

Der vorliegende Beitrag beschreibt eine Entwurfsstrategie sowie wesentliche Grundsätze, die beim Entwurf eines integrierten technischen Informationssystems zu berücksichtigen sind. Die Strategie betont den formalen Ablauf sowie die Notwendigkeit, mehrere Methoden simultan anzuwenden. Dabei soll der Rechner den Entwickler eher unterstützen als kontrollieren.

Le présent article décrit une stratégie de projet ainsi que les principes majeurs devant être pris en compte lors de l'élaboration d'un système d'information technique intégré. La stratégie souligne le déroulement formel ainsi que la nécessité d'appliquer simultanément plusieurs méthodes. Au cours de cette activité l'ordinateur est censé soutenir l'opérateur plutôt que de le contrôler.

Adresse des Autors

Dr. *David Goudie*, Leiter Sparte Netzmanagement, Colenco AG, Parkstrasse 27, 5401 Baden.

Ein Informationssystem für Elektrische Energieversorgungsunternehmen (EVU) muss durch die Bereitstellung von Informationen und Funktionen die operationellen Abläufe des Unternehmens unterstützen. Da die im Informationssystem gespeicherten Informationen sowohl strategischer als auch operationeller Art sein können, hat ihre Sicherheit und Integrität beim Systementwurf erste Priorität. Das System sollte nicht primär auf den gelegentlichen, ungeschulten Benutzer ausgerichtet sein; dennoch muss der innere Aufbau und die Konfiguration des Systems für den Benutzer transparent sein.

In einem integrierten Informationssystem dürfen Benutzerinteraktion und Datenzugang nicht durch technische Eigenschaften einzelner Systemkomponenten, sondern ausschliesslich durch unterschiedliche Datentypen und unterschiedliche Funktionalität begrenzt sein – soweit unterschiedliche Datenformate nicht konvertierbar sind. Konkret bedeutet dies etwa, dass das System die Möglichkeit bieten muss, Texte, Graphiken und Bilder frei in eine Dokumentation einzubauen, dass aber eine automatische Übersetzung von Texten oder die automatische Erstellung von Bildlegenden nicht gefordert werden können. Eine explizite Forderung an ein Informationssystem besagt ferner, dass – abgesehen von Sicherheitsüberlegungen – jedes einzelne Datenelement nur einmal und nur in einer Form gespeichert werden soll.

Ein technisches Informationssystem enthält in erster Linie Angaben über die physikalischen Eigenschaften von «Dingen» oder Objekten. Beim heutigen Stand der Technik sind im allgemeinen unterschiedliche Modelle und Methoden erforderlich, um unterschiedliche Eigenschaften dieser Objekte zu beschreiben. Im Rahmen auf Seite 15 findet sich ein Überblick über die Vielfalt der Datentypen, die ein integriertes technisches Informationssystem beherrschen muss.

Es mag etwas überraschen, dass integrierte Systeme nicht dadurch realisiert werden, dass man alle Komponenten zu vereinheitlichen versucht, sondern indem man die vom System zu akzeptierenden Datentypen identifiziert, formell definiert und anschliessend die vorhandenen Daten gemäss diesen Definitionen klassifiziert.

Das System als Entwurfsziel

Das Entwurfsziel ist ein integriertes technisches Informations-Management-System, das online die für Entwurf, Beschaffung und Unterhalt erforderlichen technischen, operationellen und wirtschaftlichen Daten des Versorgungsnetzes enthält und das weiterhin auch die erforderlichen Funktionen für Berichterstattung, Analyse, Entwurf, Planung und Betrieb bereitstellt. Information kann sich dabei auf numerische Daten, auf Texte, Graphiken, Diagramme und Bilder beziehen und zwar sowohl bezüglich des aktuellen Zustandes des Systems als auch bezüglich Planungsszenarien oder statistischer Analysen. Es müssen deshalb fortschrittliche Darstellungs-, Melde- und Analyse-Algorithmen verfügbar sein. Die Datenbank des Systems muss ausdrücklich sämtliche vorhersehbaren geographischen, betrieblichen und organisatorischen Strukturen des EVU sowie Verbindungen und Charakteristika aller Einrichtungen berücksichtigen. Sämtliche Daten sind bezüglich Herkunft, Umfang und Qualität zu qualifizieren. Diese Grundsätze sind auch bei der Implementierung eines umfangmässig nicht sehr grossen Systems zu berücksichtigen.

Ein integriertes technisches Informationssystem lässt sich allein schon durch die dadurch gegebene höhere Datensicherheit und -zuverlässigkeit sowie die Kostenvorteile einer gut definierten und korrekten Datenbank rechtfertigen. Unkoordiniertes Wuchern von Datenbeständen innerhalb eines EVU verursacht bereits an sich enorme Kosten. Zusätzlich sind jedoch noch jene Kosten zu berücksichtigen, die entstehen, wenn Analysen und Entscheide auf falschen Daten beruhen.

Systementwurf - alles ist Entwurf, der Entwurf ist alles

Der Entwurf eines Informatiksystems umfasst den gesamten Prozess von der Abstraktionsebene einer Idee bis zu jener Ebene, die sich zur Ausführung auf dem Rechner eignet. Im Laufe des Entwurfsprozesses werden viele Aspekte des Entwurfes wiederholt verändert, korrigiert oder erweitert. Erfolgreiche, elegante Systeme sind meist das Resultat eines solchen iterativen Prozesses.

Der Entwurf kann zweckmässigerweise durch eine (grosse) Zahl von Einheiten - Module oder Abschnitte - beschrieben werden. Der Quelltext dieser Einheiten sollte in einer sehr flexiblen Kombination von formaler Sprache und freiem Text geschrieben werden. Die zahlreichen Abhängigkeiten im Systementwurf müssen dabei ausdrücklich festgehalten werden; sie dürfen nicht stillschweigend, implizite in die Einheiten hineingedacht werden. Die meisten Systementwürfe sind von Natur aus stark vernetzt, und eine Beschränkung auf eine willkürlich vereinfachte Struktur ist deshalb nicht zweckmässig. Die Quellen müssen zusätzliche Angaben enthalten, die eine Darstellung des Entwurfes in verschiedenen geeigneten sicht- oder lesbaren Formen erlauben [1]. Case[1] Tools dienen primär zur Unterstützung des Entwurfsvorganges. Sie verfügen über geeignete Ablagemethoden, sie erleichtern die Systemgenerierung aus dem Quelltext und ermöglichen die Verfolgung und Analyse des Entwurfes. Case Tools können jedoch die kreative Phase des Entwurfes nicht direkt unterstützen oder überwachen.

Viele Misserfolge im Systementwurf sind auf mangelnde Disziplin und mangelnden Formalismus in der Entwurfs- und Realisierungsphase sowie auf eine unzulängliche konzeptionelle Basis zur Integration unterschiedlicher Datentypen und unterschiedlicher Funktionalität zurückzuführen.

Objektorientierter Entwurf

Die Technik des objektorientierten Entwurfes (OODT - Object Oriented Design Techniques [2]) ist für den Entwurf integrierter technischer Informationssysteme besonders geeignet, da sich diese definitionsgemäss mit komplexen physikalischen Objekten befassen. Ein objektorientierter Entwurf hat heute gegenüber einem traditionellen, auf der Funktionalität basierenden Entwurf (Top-Down Functional Design) bedeutende Vorteile, nicht zuletzt weil er den Transfer des Entwurfes von der Beschreibung in einer natürlichen Sprache in eine formale Rechnersprache unterstützt.

Ein objektorientiertes Vorgehen führt zu einem stark strukturierten Entwurf, und seine Implementierung kann, Hand in Hand, von einfach generisch[2] zu detailliert komplex-spezifisch erfolgen. Der Entwurf ist gegenüber funktionalen Anforderungen, späteren Erweiterungen und funktioneller Implementierung relativ unempfindlich; die eher generischen Ebenen können darüberhinaus als operationelle Prototypen betrachtet werden, auf denen die eher spezifischen Ebenen aufbauen, ohne diese zu ersetzen. Wichtig ist die Feststellung, dass die generischen Routinen, falls sie richtig entworfen wurden, über die Daten und die Funktionalität der abgeleiteten Objekte «wissen, was sie wissen müssen» und dass die höheren Routinen, die die Charakteristika der generischen Routinen «erben», vollständig konsistent zu den niedrigeren Routinen ablaufen.

Die Unterstützung des «Prototyping» durch einen objektorientierten Entwurf ist insofern von Bedeutung, als in einem gewissen Sinn jede Systemversion als Prototyp für die nachfolgende Version aufgefasst werden kann. Auch wenn spätere Erweiterungen Änderungen oder die Zufügung neuer Basisobjekte erfordern können, wird die zugehörige Funktionalität entweder automatisch durch das System propagiert oder sie kann eingegliedert werden. Das bedeutet, dass das System erweitert werden kann, um die neue Funktionalität zu unterstützen, unabhängig von deren vollen Implementierung.

Erweiterung des objektorientierten Entwurfes auf die Systemebene

Objektorientierte Techniken können auf natürliche Weise auf die Systemebene erweitert werden, vorausgesetzt, man akzeptiert die Notwendigkeit einer Mehrzahl verschiedener Paradigmen[3] oder formaler Sprachen innerhalb des Systems [3;4;5].

Ein früher, bedeutender Schritt in der Anwendung objektorientierter Techniken wurde vor mehr als zwanzig Jahren durch die Entwicklung der Unix Make Utility[4] vollzogen. Dies war insofern ein wesentlicher Durchbruch im Bereich des Software Engineering, als die Notwendigkeit erkannt wurde, den ausführbaren Prozess, respektive seine Generierung, zu definieren, eine formale Sprache für die Definition zu verwenden und mit der Make Utility die Definition von der Realisierung zu trennen. Leider scheint der Gebrauch solcher Techniken auch heute noch wenig verbreitet zu sein.

Formale Sprachen

Es wurde bereits darauf hingewiesen, dass der Entwurf auf der höchstmöglichen Ebene und in einer möglichst präzisen Form formuliert werden soll, weil diese Form die Absichten des Entwicklers am besten wiederzugeben vermag und gegenüber Änderungen während der Realisierung am unempfindlichsten sein sollte. Dies kann wohl am besten - und vielleicht ausschliesslich - durch den Gebrauch formaler, lesbarer Sprachen erreicht werden. Abgesehen von den gut bekannten Programmiersprachen sind die Preprozessorsprache von C, die Unix Make Utility, Knuths TeX, Adobes Post Script und Ashton Tates dBase-Konfigurationssprache (Set-Befehle) ausgezeichnete Beispiele für den Nutzen solcher Methoden.

Als spezifisches Beispiel können wir annehmen, es sei zur Dateneingabe eine Maske entworfen worden und es sei nun deren Hintergrundfarbe festzu-

[1] Computer Aided Software Engineering

[2] allgemein

[3] Darstellungen

[4] Make ist ein Programm, welches die Generierung eines Programmsystems ausgehend von einer Definitionsdatei erlaubt.

legen. Bei manchen Systemen kann der Benützer nach Belieben ein Icon oder eine Menu-Position - etwa mit Hintergrundfarbe bezeichnet - anwählen und dann in einem nachfolgenden Menu beispielsweise die Farbe Blau wählen. Das System akzeptiert diese Spezifikation und speichert die Information üblicherweise in einer «versteckten» Form, beispielsweise durch eine Modifikation der ausführbaren Bilddatei, durch Speicherung der Information in einer zusätzlichen Datenbank, durch Setzen einer bestimmten Variablen in einer Konfigurationsdatei auf einen bestimmten Wert oder, wie hier vorgeschlagen, durch die Verwendung einer formalen Sprache, d.h. durch die Eingabe von Text - etwa «Setze Hintergrundfarbe auf Blau» - in einer «sichtbaren» Konfigurationsdatei.

Abgesehen von der Unempfindlichkeit sprachlicher Formen gegen Verfälschungen fällt es relativ leicht, einen angemessenen Übersetzungsmechanismus zwischen unterschiedlichen formalen Sprachen bereitzustellen.

Grundsätze für einen integrierten technischen Systementwurf

Ein integriertes System muss mehr sein als eine Ansammlung von Komponenten, die im Laufe der Zeit ohne Disziplin und schlecht dokumentiert, das heisst nicht nachvollziehbar, auf einem Rechner oder auf einem Rechnernetz abgelegt wurden. Die Grundlage eines integrierten Systems ist eine präzise formale Definition. Folgende Grundsätze sollten beim Entwurf eines integrierten technischen Informationssystems befolgt oder mindestens in Erwägung gezogen werden:

1. Das Vorgehen soll so diszipliniert und formal als möglich sein.

2. Man versuche, das System zu modularisieren und die Spezifikationen und Definitionen von der Realisierung zu trennen, so dass ein möglichst minimaler Informationstransfer zwischen den am Projekt Beteiligten notwendig ist. Diskussionen sollten auf zwei Fragen beschränkt bleiben: «Entspricht die Definition dem, was wir wollen; ist sie vollständig?» und «Entspricht das System der Definition?».

3. Man verwende eine formale Struktur mit integrierten freien und formalen Textteilen. Für die Teile, die vom Rechner interpretiert werden müssen, werden mehrere Paradigmen erforderlich sein. Grosses Gewicht sollte selbst dann auf eine formale oder pseudoformale Syntax gelegt werden, wenn keine automatische Interpretation möglich ist.

4. Systemelemente sollen von den einfach-generischen zu den detailliert-komplex-spezifischen geplant und entworfen werden. Spezielle Sorgfalt ist der Festlegung der Schnittstellen zu widmen.

5. Der Systementwurf soll in Ebenen erfolgen, und es ist sicherzustellen, dass jede Ebene ausschliesslich eine strukturierte Objektmenge enthält, dass sie die Funktionalität bereitstellt, um diese Objekte als Ganzes zu manipulieren und sie frei zu bewegen oder um ein bestimmtes Objekt anzuwählen. Funktionalität, die erforderlich ist, um die interne Struktur eines Objektes zu manipulieren, muss durch dieses und nur durch dieses Objekt geliefert werden.

6. Man mache von der Tatsache Gebrauch, dass mit der Realisierung des Top-Level-Entwurfes eigentlich ein operationeller Prototyp entsteht, und man stelle sicher, dass Systementwurf, Systemrealisierung und System-Funktionstest Hand in Hand ablaufen.

7. Objekt- oder datenorientierte Entwurfsmethoden sind zu bevorzugen, denn Methoden für die formale Definition von Datenstrukturen sind wesentlich weiter fortgeschritten als die Methoden für die Beschreibung der Funktionalität. Beim Systementwurf sind Entscheidungen bezüglich der Datenstrukturen meist die kritischsten. Die vielen Vorteile, die dadurch entstehen, dass ein System auf Objekten aufbaut und Funktionalität auf diese Objekte angewendet wird, sind gut dokumentiert.

8. Man betrachte die Prozeduren für die Systemgenerierung und die Konfigurationsparameter als Teil des Systementwurfes. Der Generierungsprozess sollte voll automatisiert sein, um sicherzustellen, dass das erzeugte System die Systemspezifikationen zuverlässig wiedergibt. Alle manuellen Operationen und Dateneingaben sollten formal dokumentiert werden. Diese Forderung widerspricht dem heute populären interaktiven, «undisziplinierten» Trend.

Datentypen eines integrierten technischen Informationssystems für EVU

Die Daten eines technischen Informationssystem können entsprechend den folgenden fünf Typen klassiert werden. Die fünf Klassierungen werden als erste Ebene der Verfeinerung in der Entwicklung von spezifischen, optimierten Strukturen angesehen, die den effizienten Zugang zu technischen Daten unterstützen.

1. Physikalische Daten: die reale, in Vektorform abgespeicherte Eins-zu-Eins-Darstellung der dreidimensionalen Umwelt. Im Extremfall kann aus einer solchen Repräsentation irgendeine Darstellung, von irgendeinem Gesichtswinkel aus betrachtet, erzeugt werden. Auch ein Ausschnitt der physikalischen Dateneinheit ist für sich betrachtet noch sinnvoll. Eine Karte stellt einen zweidimensionalen Fall dar. Zooming und Panning[5] sind zulässig.

2. Symbolische Daten: schematische Darstellungen, beispielsweise Schaltdiagramme als Abbild eines Systems der realen Welt. Sie sind im wesentlichen zweidimensional, jedoch ohne massstabgerechten Bezug zur Wirklichkeit. Sie erfordern einen diskreten Ansatz beim «Zooming» (Skalierung mit Decluttering[6]). Auch bei symbolischen Daten kann die Betrachtung eines Teils sinnvoll sein.

3. Stark strukturierte Daten: Darstellung und Speicherung von Daten aufgrund eines logischen Schemas, wobei die Interpretation der Daten ohne dieses Schema nicht möglich ist. Relationale Datenbanken und Spreadsheets fallen in diese Kategorie.

4. Minimal strukturierte Daten: Speicherung von Textinformation und zugehörigen Kontrollsequenzen für die Darstellung entsprechend vordefinierter Formatierungskonzepte. Von speziellen Fällen abgesehen sind Zooming und im allgemeinen auch Ausschnitte nicht sinnvoll. Man beachte, dass innerhalb gewisser Grenzen das Layout von Text und Zahlen auf deren Bedeutung keinen Einfluss hat.

5. Bilder und Videodaten: Sie werden aufgrund ihrer speziellen Eigenschaften als gesonderte Kategorie betrachtet. Bilder werden als statisch, Videodaten als dynamisch betrachtet. Sie sind in Pixelform gespeichert und können, abgesehen von sehr speziellen Ausnahmen, durch den Rechner nicht interpretiert werden.

[5] Überstreichen eines Winkelbereiches
[6] Entfernen von Detailinformationen

9. Für wichtige Schnittstellen verwende man Standardsprachen wie SQL, Post Script und X-Windows, oder man entwerfe für die wichtigsten Schnittstellen speziell angepasste Sprachen.

10. Durch Pseudo Reverse Engineering [6] schliesse man Entwurfsdefinitionen für vollimportierte Komponenten ein.

11. Man beachte strikte die Kontrollprozeduren für Arbeits- und Freigabeversionen.

12. Man versuche, mindestens auf dem Top-Level, eine Entwurfsphilosophie zu verwenden, die jener, die bei der Realisierung des Systems verwendet wird, um eine Generation voraus ist.

Anwendung dieser Entwurfstechnik auf die Erstellung von integrierten Systemen unter Verwendung von Standardpaketen

Soll ein System vorwiegend durch Standard- oder vollständig importierte Pakete realisiert werden, sollten die vorgehend beschriebenen Methoden sowohl bei der Auswahl der Pakete als auch bei der Realisierung des Systems mittels dieser Pakete eingesetzt werden. Selbst wenn der Umfang der automatischen Realisierung gering sein mag, kann die vorgeschlagene Methodik unter Einsatz manueller Konversion der formalen Definition bei der Implementierung immer noch Vorteile bieten.

Bei den meisten Systemen liegt wohl der grösste Entwurfsaufwand der Entwickler oder des EVU-Personals in der Datenbank und in den damit verbundenen Dateneingabe- und Berichtsprogrammen. Hier ist die Anwendung der beschriebenen Methoden besonders wichtig, auch wenn die Verkaufspublikationen der Datenbankhersteller bezüglich der erforderlichen Disziplin und der formalen Techniken ab und zu einen andern Eindruck vermitteln. In Übereinstimmung mit den aufgestellten Grundsätzen muss die Datenbankentwurfsdefinition alle Aspekte des Datenbankentwurfes berücksichtigen; sie kann sich nicht auf jene beschränken, die durch die aktuelle Version des gewählten Datenbank-Management-Systems akzeptiert werden. Die Datenbank-Entwurfsdefinitionen sollten ebenfalls auf der höchstmöglichen Ebene und so formal wie möglich formuliert werden. Entitäten, Beziehungen zwischen Entitäten, Domänen, Schlüssel usw., aber auch wesentliche Insert-, Delete- und Update-Regeln sind explizite zu definieren und zu dokumentieren. Darüberhinaus ist es unerlässlich, dass EVU, die bezüglich ihrer betrieblichen Abläufe ja zunehmend auf Rechnerunterstützung zurückgreifen, diese Abläufe korrekt und formal definieren sowie implementieren und dass das Management diese Festschreibung unterstützt. Auch hier sei wiederum die Trennung zwischen der formalen Festlegung dessen, was erreicht werden soll und der Implementierung betont.

Schlussfolgerung

Es wurde das Prinzip aufgezeigt, wonach der dokumentierte Entwurf eines integrierten technischen Informationssystems explizite und formal alles Notwendige enthalten muss, um ein solches System zu produzieren, dass der Rechner den Entwickler beim Verfolgen und Kontrollieren des Entwurfes unterstützen muss, dass er aber die kreative Phase nicht behindern darf. Eine gesunde Mischung von formalem und freiem Text dürfte dieser Forderung am besten entsprechen. Es wurde schliesslich betont, dass das generierte, operationelle System den formalen, dokumentierten Entwurf wieddergeben muss, und nicht lediglich das Resultat zahlreicher interaktiver, durch nachträglich nicht identifizierbare Personen ausgeführter Abläufe am Terminal sein darf.

Literaturverzeichnis

[1] *P.W. Oman* and *C.R. Cook:* The Book Paradigm for Improved Maintenance. IEEE Software, January 1990, pp 39...45.

[2] *B. Meyer:* Objectoriented Software Construction. Prentice-Hall 1988.

[3] *A.J. Czuchy, Jr.* and *D.R. Harris:* KBRA: A new Paradigm for Requirements Engineering. IEEE Expert 3(1988)4, pp 21...35.

[4] *G.E. Kaiser, N.S. Barghouti, P.H. Feiler & R.Schwanke:* Database Support for Knowledge-Based Engineering Environments. IEEE Expert 3(1988)2, pp 18..32.

[5] *S. Rugaber, S.B. Ornburn* and *R.J. Le Blanc:* Recognising Design Decisions in Programs. IEEE Software January 1990, pp 46...54

[6] *E.J. Chikofsky* and *J.H. Cross:* Reverse Engineering and Design Recovery: A Taxonomy. IEEE Software, January 1900, pp 13...17