Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 76 (1985)

Heft: 17

Artikel: Das Arbeitsfeld der Softwaremacher

Autor: Bernhard, D.

DOI: https://doi.org/10.5169/seals-904670

Nutzungsbedingungen

Die ETH-Bibliothek ist die Anbieterin der digitalisierten Zeitschriften auf E-Periodica. Sie besitzt keine Urheberrechte an den Zeitschriften und ist nicht verantwortlich für deren Inhalte. Die Rechte liegen in der Regel bei den Herausgebern beziehungsweise den externen Rechteinhabern. Das Veröffentlichen von Bildern in Print- und Online-Publikationen sowie auf Social Media-Kanälen oder Webseiten ist nur mit vorheriger Genehmigung der Rechteinhaber erlaubt. Mehr erfahren

Conditions d'utilisation

L'ETH Library est le fournisseur des revues numérisées. Elle ne détient aucun droit d'auteur sur les revues et n'est pas responsable de leur contenu. En règle générale, les droits sont détenus par les éditeurs ou les détenteurs de droits externes. La reproduction d'images dans des publications imprimées ou en ligne ainsi que sur des canaux de médias sociaux ou des sites web n'est autorisée qu'avec l'accord préalable des détenteurs des droits. En savoir plus

Terms of use

The ETH Library is the provider of the digitised journals. It does not own any copyrights to the journals and is not responsible for their content. The rights usually lie with the publishers or the external rights holders. Publishing images in print and online publications, as well as on social media channels or websites, is only permitted with the prior consent of the rights holders. Find out more

Download PDF: 12.07.2025

ETH-Bibliothek Zürich, E-Periodica, https://www.e-periodica.ch

Das Arbeitsfeld der Softwaremacher

D. Bernhard

Der Aufsatz beschreibt den Ablauf eines Softwareprojektes sowie die Aufgaben und Tätigkeiten des Softwareingenieurs. Er will dem Leser ein realistisches Bild der heutigen Situation, insbesondere im Bereich der technischen Software, vermitteln. Der abschliessende Ausblick zeigt, in welcher Richtung der zukünftige Entwicklungsgang zu erwarten ist.

L'article décrit le déroulement d'un projet de logiciel et les diverses activités d'un ingénieur chargé du développement. Le lecteur peut ainsi se faire une idée concrète de la situation telle qu'elle se présente actuellement, en particulier dans le domaine du logiciel technique. Les changements de tendances que le futur nous réserve sont indiqués dans la dernière partie de cet article.

Adresse des Autors

D. Bernhard, RETIS Realtime Software AG, Bahnhofstrasse 96, 5001 Aarau.

1. Einleitung

Die Nachfrage nach produktivitätssteigernder Computertechnologie wächst sehr rasch. Den Engpass stellt dabei mehr und mehr die Software dar. Während sich bei der Hardware das Preis-/Leistungs-Verhältnis innert 5 Jahren in der Grössenordnung um einen Faktor 10 verbessert, geht dieser Prozess bei der Softwareerstellung wesentlich langsamer vor sich. Dies führt dazu, dass der Anteil der Softwarekosten verglichen mit den Hardwarekosten laufend zunimmt. Bei industriellen Anwendungen ist mittlerweile das Verhältnis so, dass die Investition in die Software im allgemeinen grösser als in die Computerhardware ist. Dadurch rückt eine Berufsgruppe vermehrt ins Zentrum des Interesses: die Softwaremacher.

Ein Ingenieur, welcher ein Produkt entwickelt, macht im Grunde genommen nichts anderes, als Informationen zu verarbeiten, in Form von Zeichnungen, Zahlenmaterial, Text und Bildern. Das gleiche macht auch der Softwareingenieur. Und das Hilfsmittel dazu ist mehr und mehr der Computer. Der Arbeitsplatz der beiden wird sich deshalb in Zukunft mehr und mehr gleichen: eine Workstation mit Bildschirm und Tastatur, welche ihnen den Zugriff zur benötigten Computerleistung gibt und über die sie auch mit anderen Benützern verkehren können. Später wird die Möglichkeit der Spracheingabe dazukommen. Auch die Anforderungen an den Produktentwickler und an den Softwareingenieur werden sich teilweise überdekken. Beide müssen in der Lage sein, ein Problem zu verstehen, es zu strukturieren und dafür kreativ Lösungen zu entwickeln. Die Unterschiede liegen jedoch im fachspezifischen Wissen, welches zur Erfüllung der speziellen Aufgaben nach wie vor erforderlich ist.

Die Zukunft wird im Bürobereich grosse Veränderungen bringen. Und hier liegt der grosse Vorteil des Softwareingenieurs, nämlich dass er an Veränderungen in seinem Arbeitsumfeld gewöhnt ist.

2. Der Ablauf eines Projekts

Noch immer glauben viele Aussenstehende, dass es genügt, eine Programmiersprache zu erlernen, um sich dann hinzusetzen und dem Computer die nötigen Anweisungen zu geben. Richtig ist jedoch, dass Software auf transparente Art und Weise anhand eines methodischen Vorgehenskonzeptes realisiert werden muss. Softwareprojekte werden geplant, zielgerichtet realisiert und anhand klar definierter Meilensteine überwacht.

Ein bewährtes Modell hierzu ist das Phasenkonzept, wie es in Figur 1 in grober Form dargestellt ist. Nach einer allgemeinen Aufgabenbeschreibung im Rahmen der Vorabklärungen erfolgt als erstes die Erstellung des Pflichtenhefts. Hierbei werden in enger Zusammenarbeit mit dem zukünftigen Benützer die Anforderungen an das System definiert (Fig. 2). Das *Was* steht dabei im Vordergrund.

Auf der Basis dieses Pflichtenhefts kann nun die Software entworfen werden, zuerst das System als Ganzes, dann die einzelnen Bausteine. Das *Wie* wird hier festgelegt. Erst nach Entwurf der Softwarestruktur wird der Programmcode geschrieben und ausgetestet (Fig. 3).

Bei der Inbetriebnahme muss die Software des gesamten Systems zusammengefügt werden (Fig. 4). Danach muss die von Anfang an geführte Dokumentation noch komplettiert werden. Dies im Hinblick auf eine spätere Wartung (Fig. 5) in Form von Feh-

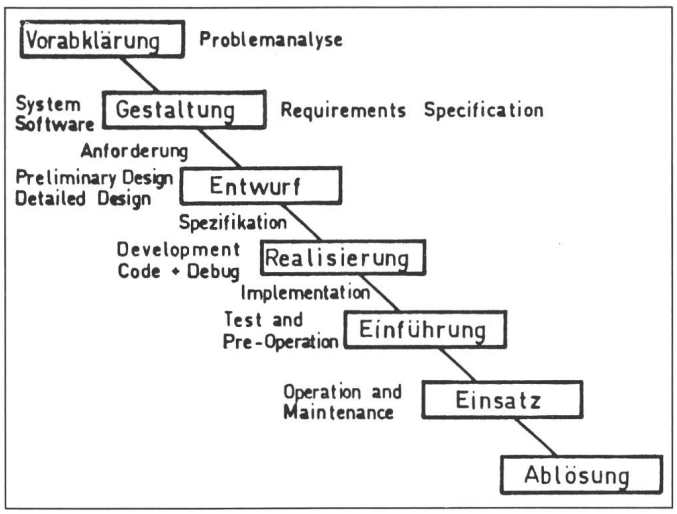

Fig. 1
Das Phasenkonzept, ein methodisches Vorgehensmodell zur Projektabwicklung

lerkorrektur, Ergänzung oder Erweiterung des Systems.

... in der Praxis

Der saubere Projektablauf, bei dem Schritt auf Schritt aufbaut, ist in der Praxis kaum anzutreffen. Vielmehr ist es so, dass die einzelnen Schritte gegenseitig überlappen oder dass von einer späteren Phase wieder in eine frühere zurückgesprungen werden muss. Die Gründe dafür sind vielfältig: Die Anforderungen an das System können sich im Verlaufe des Projekts ändern, der Zeitdruck kann ein überlapptes Arbeiten erforderlich machen, einzelne Beteiligte sind mit ihren Arbeiten im Rückstand (Fig. 6), oder es können Probleme bei der technischen Realisierung auftreten, welche eine Änderung des Konzeptes erzwingen. Gerade der auf der Seite des Benützers zu leistende Aufwand wird häufig unterschätzt. Auch ist es so, dass wegen der neuen Möglichkeiten stets komplexere Aufgaben in Angriff genommen werden. Der praktische Ablauf ist deshalb vielschichtig und dynamisch.

Als Vorgehensmodell und Orientierungshilfe hat sich das Phasenkonzept jedoch gut bewährt. Es ist sicher besser, eine Planung zu haben, welche rollend der sich ändernden Situation angepasst wird, als wegen der auftretenden Änderungen auf eine Planung zu verzichten und sich treiben zu lassen.

3. Aufgaben und Tätigkeiten des Softwareingenieurs

Die obige Beschreibung des Projektablaufs zeigt, dass der Softwareingenieur nur einen kleinen Teil seiner Zeit mit dem Eintippen von Programmcodes verbringt. Er muss ja zuerst die Aufgabenstellung verstehen und beschreiben, dann das System entwerfen, genügend dokumentieren und es am Schluss auch austesten.

In Projekten mit mehreren Mitarbeitern wird die Arbeit aufgeteilt. Dabei bewährt sich eine maximale Teamgrösse von 6 bis 8 Mitarbeitern, grössere Projekte werden mit Vorteil in mehrere Teilprojekte aufgeteilt. Die Teamgrösse variiert im Verlauf des Projekts. Ein erfahrener Mitarbeiter, meist zugleich Projektleiter, erarbeitet Pflichtenheft und Spezifikation. Für diese Aufgabe muss er gut mit dem Kunden, der meist auch zukünftiger Benützer des Systems ist, kommunizieren können, um dessen Bedürfnisse und Anforderungen richtig zu verstehen.

Anschliessend hat er diese Anforderungen schriftlich niederzulegen. Die Hilfsmittel dazu sind dabei nach wie vor der Texteditor, welcher ein einfaches Ändern und Ergänzen des einmal erstellten Textes erlaubt, sowie Papier, Bleistift und Radiergummi. Wichtig ist die Erkenntnis, dass die Denkarbeit wohl sehr wichtig ist, aber erst als Resultat gelten kann, wenn sie ihren Niederschlag auf Papier (oder einem elektronischen Informationsträger) gefunden hat.

Zum Entwurf der Software wird im allgemeinen das Team erweitert. Die Aufgabe wird in Teilgebiete unterteilt, deren Schnittstellen (ein schöneres Wort heisst: Nahtstellen) zuerst festgelegt werden müssen, bevor die einzelnen Teilgebiete unabhängig voneinander bearbeitet werden können. In dieser schöpferischen Phase stehen dem Softwareingenieur zur Darstellung der gewählten Lösungen verschiedene Techniken zur Verfügung, die bekanntesten sind *Jackson, State-Event, Mascot* und *Struktogramm*. Es gibt Hilfsmittel, welche diese Techniken unterstützen, sie sind aber noch wenig verbreitet. Das Resultat der Entwurfsphase ist die Beschreibung der zu erstellenden Software, aufgeteilt in einzelne Module.

Der Entwurf, die Codierung und das Austesten der einzelnen Programmodule kann gut auf weitere Mitarbeiter verteilt werden. Hierbei können auch jüngere Mitarbeiter Erfahrungen sammeln. Es handelt sich um Arbeiten, welche weitgehend am Computer erfolgen. Der Entwurf kann im Editor dokumentiert werden. Das Erstellen des Programmcodes sowie das Compilieren und Austesten geschehen heute fast ausschliesslich durch den Programmierer direkt am Computer. Wichtig bei diesen Arbeiten ist eine klare, strukturierte Denkweise sowie ein diszipliniertes Vorgehen. Der erstellte Code soll so strukturiert und dokumentiert sein, dass er sowohl von seinem Ersteller als auch von einem Aussenstehenden später gut verstanden werden kann.

Bei der Einführung des Systems wird das Softwareteam im allgemeinen wieder verkleinert. Es geht nun darum, Mängel der Software im Zusammenspiel mit dem gesamten System zu lokalisieren und zu beheben. Diese Arbeiten erfolgen häufig unter Zeitdruck; dies verlangt von den Beteiligten, dass sie auch unter erschwerenden Bedingungen Ruhe und Übersicht bewahren. Da die auftretenden Fehler nicht vorhersehbar sind, kann diese Arbeit kaum formalisiert werden. Die Korrekturen müssen trotzdem diszipliniert und mittels wiederholbarer Testverfahren erfolgen; nur so lassen sich neue Fehler vermeiden.

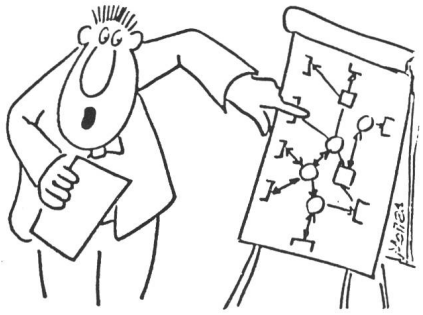

Fig. 2 Pflichtenheft

4. Bedeutung des Projektmanagements

Die Erfahrung zeigt, dass Probleme wie massive Kosten- und Terminüberschreitungen oder unbefriedigende Lösungen häufig auf Mängel im Projektmanagement zurückzuführen sind. Ein gutes Projektmanagement ist deshalb Voraussetzung für die erfolgreiche Abwicklung eines Softwareprojekts. Das Projektmanagement beginnt bei der Projektorganisation. Dabei sollen die zukünftigen Benützer, Fachpersonal und Management, eingehend ins Projekt miteinbezogen werden. Die wesentlichen Aufgaben des Projektmanagers sind die Schätzung des Aufwands, die Planung der Arbeiten, die Führung der Mitarbeiter sowie die Kontrolle des Projektfortschritts und des Aufwands, wobei die Schätzung des Aufwands besondere Schwierigkeiten bietet. Einerseits handelt es sich bei der Softwareerstellung um schwer formalisierbare Arbeiten, anderseits ist auch die Problemstellung jedesmal neu, da kaum jemand das gleiche Programm zweimal schreibt.

Fig. 3 Softwarestruktur

Gute Software-Projektmanager sind rar und die Anforderungen an sie recht hoch. Der Projektmanager soll sich auf den Gebieten Menschenführung, Planung und Software auskennen, und zwar ziemlich gut, denn die technischen Problemstellungen sind recht schwierig und die Mitarbeiter im allgemeinen eher anspruchsvoll. Aus diesem Grunde wird bei grösseren Projekten manchmal ein zusätzlicher administrativer Projektleiter eingesetzt, welcher den Softwareleuten die Planungs- und Projektadministrationsaufgaben mindestens teilweise abnimmt.

5. Methoden und Hilfsmittel

Der Softwareingenieur steht allen elektronischen Hilfsmitteln, welche seine Arbeit erleichtern, naturgemäss positiv gegenüber. Das am häufigsten benützte Software-Hilfsmittel ist der Editor, der zur Erstellung der Dokumentation und zur Eingabe der Programme eingesetzt wird. Mit seiner Hilfe lassen sich Dokumente sehr einfach ändern oder ergänzen. In dieser Möglichkeit liegen allerdings auch Gefahren, denn wo leicht geändert werden kann, wird gerne auch leichtsinnig geändert. Hier helfen Systeme zur Verwaltung der verschiedenen Versionen, vergleichbar mit dem technischen Änderungswesen in einer Produktionsfirma.

Die Bedürfnisse des Softwareingenieurs gehen jedoch weiter. Er möchte auch Zeichnungen editieren, er braucht eine Projektplanung, er möchte aus gezeichneten Strukturen möglichst automatisch zum Programm kommen. Bereits seit einiger Zeit sind verschiedene integrierte Entwicklungssysteme erhältlich, welche den Softwareingenieur von der Pflichtenhefterstellung über den Entwurf bis zur Codierung, Projektleitung und Dokumentation unterstützen. Ihr wirtschaftlicher Einsatz insbesondere im technischen Gebiet ist jedoch heute noch sehr begrenzt. Kaum jemand zweifelt daran, dass hier noch eine grosse Entwicklung stattfinden wird.

6. Standardisierung – Traum und Wirklichkeit

Die Entwicklungsumgebung des Softwareingenieurs, die Werkzeuge und Zielsysteme, sind einem ständigen Wandel unterworfen. Dies erhält ihn zwar geistig frisch, vermindert aber auch seine Produktivität. Insbesondere ist es mit der Portabilität, d.h. der Übertragbarkeit der Software vom System eines Herstellers auf das eines anderen, nach wie vor schlecht bestellt.

Unix (TM) ist einer der Versuche, einen einheitlichen Standard zu schaffen, aber inzwischen hat sich auch hier eine Vielzahl von Versionen gebildet, wobei noch jeder Hersteller seine Eigenheiten einbaut. Der Grund für die Vielfalt dürfte letztlich in der rasanten Entwicklung bei der Hardware liegen. Eine Standardisierung lässt sich nur auf dem kleinsten gemeinsamen Nenner erreichen. Dies wiederum würde es verunmöglichen, aus dem technischen Fortschritt den entsprechenden Nutzen zu ziehen. Die Vielfalt hat jedoch ihren Preis, und diesen zahlt schliesslich der Endkunde. Bei ihm, und nicht beim Hersteller, liegt deshalb das primäre Interesse an einer Standardisierung.

Um durch diesen Dschungel hindurchzukommen, muss sich ein Softwareunternehmen auf einige strategische Betriebssysteme festlegen. Wichtige Auswahlkriterien sind dabei die Marktverbreitung sowie die Leistungsfähigkeit des Herstellers. *Retis* erstellt die meisten Anwendungen unter *VMS* oder *RSX*, Unix wird selten eingesetzt, da es kaum echtzeitfähig ist. Festzulegen sind aber auch die strategischen Programmiersprachen, bei Retis insbesondere *Pascal*, wie auch strategische Datenbanksysteme und Transaktionsmanager.

Wo vorhandene Erfahrungen weiterverwendet werden können, lässt sich die Produktivität erheblich steigern. Die Einführung eines neuen Betriebssystems belastet ein Projekt erfahrungsgemäss mit etwa 500 bis 1000 zusätzlichen Arbeitsstunden, je nach Komplexitätsgrad. Für eine neue Programmiersprache kann man mit etwa 100 Stunden je Teammitglied rechnen, und zum Kennenlernen eines neuen Programmentwicklungssystems mit etwa 50 Stunden. Am schwierigsten ist eine Vorhersage, wenn es darum geht, in einem neuen Anwendungsbereich zu arbeiten, da das Verständnis dafür eine wesentliche Know-how-Komponente bildet.

7. Bestehende Software oder Neuentwicklung?

Eine Software fertigzustellen und zum Funktionieren zu bringen bedeutet ein beträchtliches Mass an Arbeit. Wenn es aber einmal soweit ist, kann die Software einfach und beliebig oft

Fig. 4 Linking

vervielfältigt werden. Das Einfachste und Billigste wäre deshalb, eine bestehende Software unverändert zu übernehmen. Im industriellen Anwendungsbereich sind jedoch die Anforderungen oft so unterschiedlich, dass dies nicht möglich ist. Am ehesten lässt sich hier ein Baukastenkonzept realisieren, wo einzelne Programmbausteine übernommen, mit neu entwickelten ergänzt und zum gewünschten Ganzen zusammengefügt werden.

Doch auch dem stehen in der Praxis grosse Widerstände gegenüber. Zuerst einmal müsste der Softwareingenieur diese Bausteine überhaupt kennen und auf ihre Eignung prüfen, zum andern muss er bei der Integration die einzelnen Funktionen sehr genau verstehen. Dies bedingt eine wesentlich bessere Strukturierung und Dokumentation der Programme als bei der Einmalverwendung, eine Vorinvestition, welche selten getätigt wird. Schliesslich haben die Softwareleute noch einen ausgesprochenen Berufsstolz, welcher es ihnen offensichtlich verbietet, die Arbeit eines anderen zu übernehmen, denn irgendein Mangel lässt sich an jedem Programm finden.

Häufig wird zuerst die Hardware ausgewählt und erst dann die zugehörige Software entwickelt. Untersucht man dabei zuerst die Frage, welche Software bereits vorhanden ist, so kann man möglicherweise erhebliche Einsparungen an Geld und Zeit erzielen.

8. Der Sonderfall Softwareunternehmen

Retis ist eine Softwarefirma, welche sich auf technische Software spezialisiert hat. Das Anwendungsgebiet lässt sich am ehesten mit «Industrielle Automation» umschreiben und umfasst die Steuerung, Überwachung und Verwaltung von unterschiedlichsten Anlagen in der Industrie. Das Charakteristische ist hierbei, dass der Computer nicht für sich allein steht, sondern dass er mit einem physikalischen System, z.B. mit einer Maschine oder Transportanlage, in Verbindung steht.

Die Arbeitgeber eines Softwareunternehmens sind in sehr direkter Art und Weise seine Kunden. Dies erzwingt geradezu ein Eingehen auf deren Bedürfnisse. Ausserdem begünstigt dies ein kostenbewusstes Denken und ein zielstrebiges Vorgehen. Anderseits kann ein Softwareunternehmen

Fig. 5 Service

im Gegensatz zu einer internen Softwareabteilung nicht eine enge Produktestrategie festlegen, was dazu führt, dass ein breites Spektrum an Betriebssystemen und Programmiersprachen eingesetzt werden muss. Dem steht jedoch wiederum der Vorteil eines grösseren Überblicks gegenüber.

Vom Softwareingenieur verlangt ein Softwareunternehmer mehr Initiative, Beweglichkeit und unternehmerisches Denken. Dafür geniesst dieser aber eine grosse Freiheit im Handeln sowie Abwechslung in seinen Aufgaben. Die meisten Mitarbeiter z.B. von Retis haben ein klassisches Ingenieurstudium hinter sich. Die Softwarekenntnisse haben sie sich im Verlaufe der beruflichen Weiterbildung auf verschiedenste Weise angeeignet. Das Spektrum reicht dabei vom On-the-job-Training über Lieferantenkurse bis zu Tages- und Abendnachdiplomstudien. Seit einigen Jahren werden Softwareingenieure auch an den HTL und der ETH ausgebildet. Diese jüngeren Mitarbeitern bewältigen die Praxis ohne Probleme.

Ein guter Softwareingenieur kennt nicht nur Programmiersprachen und Betriebssysteme, sondern er besitzt auch ein gewisses Verständnis für das Anwendungsgebiet, in welchem seine Software schliesslich läuft. Es ist deshalb wichtig, dass der Anwender in der Pflichtenheftphase seine Anforderungen klar definiert und der Softwareingenieur diese auch versteht. Eine gute Lösung wird nur durch eine enge Zusammenarbeit zwischen Fachspezialist und Softwarespezialist gefunden.

Die ständige Weiterbildung der Mitarbeiter ist eine wichtige Aufgabe. Dies bedingt erhebliche Investitionen in Form von Kurskosten und «unproduktiver» Arbeitszeit. Dabei genügt es nicht, Kenntnisse über Betriebssysteme und Programmiersprachen zu vermitteln, sondern es sind auch Methoden für Design, Dokumentation und Projektmanagement zu schulen. Schliesslich ist auch der Nutzen des Erfahrungsaustausches unter den Mitarbeitern, welche an verschiedenen Projekten arbeiten, nicht zu unterschätzen.

9. Zukunft

Der Beruf des Softwareingenieurs ist heute tatsächlich ein Handwerk mit goldenem Boden. Kaum eine Berufsgruppe sieht sich heute einer so grossen Nachfrage gegenüber. Doch was wird in 5 oder 10 Jahren sein? Werden die verschiedenen Schulen bis dann einen Überfluss an solchen Spezialisten erzeugt haben? Oder werden gar die Entwicklungen im Bereich der künstlichen Intelligenz den Programmierer ersetzen und damit überflüssig machen?

Im Moment kann man davon ausgehen, dass die Entwicklung im Bereich der Hardware auch in den nächsten Jahren mit unverminderter Geschwindigkeit weitergehen wird. Dies führt dazu, dass bei der Softwareentwicklung immer weniger Rücksicht auf den Verbrauch der Hardwareressourcen genommen werden muss. Dadurch wiederum wird es möglich, dem Entwickler komfortablere Werkzeuge zur Verfügung zu stellen, welche einen Teil seiner heutigen Routineaufgaben übernehmen.

Bereits heute gibt es Programmentwicklungssysteme, die erlauben, in sehr kurzer Zeit einen funktionsfähigen Prototyp für eine bestimmte Anwendung zu erstellen, vor allem was den Dialog und die Datenbank anbetrifft. Dieser kann dann vom Benützer ausgetestet werden, bevor die Anwendung in ihrer endgültigen Form erstellt wird. Wesentliche Hilfen werden auch integrierte Programmentwicklungssysteme bringen, welche den Softwareingenieur von der Problemstellung über die Realisierung bis zur Wartung unterstützen. Hauptsächliche Hilfsfunktionen sind dabei ein Editor, welcher Text, Schemas und Bilder gleichermassen verarbeiten kann, ein Ablagesystem, welches die erstellten Versionen selbständig verwaltet, ein Projektplanungs- und Kontrollsystem sowie die Umwandlung der beschriebenen Aufgabe in den Maschinencode auf einer sehr hohen Ebene.

Eine weitere wichtige Entwicklung ist im Bereich der Endbenützersprachen zu erwarten. Mit problemorien-

tierten Programmiersprachen wird es möglich sein, dass der Fachspezialist seine Aufgabenstellung ohne Umweg über den Programmierer direkt dem Computer eingibt. Bereits heute sind verschiedene solche Systeme erhältlich, welche besonders zum Erstellen einfacher Dateien und Graphiken benützt werden. Die Entwicklung geht jedoch weiter, und auch die künstliche Intelligenz wird dem Benutzer die Bedienung solcher Systeme weiter erleichtern. Ohne eine gewisse EDV-spezifische Ausbildung geht es jedoch auch hier vorläufig nicht. Dem kommt aber der heutige Trend entgegen, diese in die Berufsausbildung generell zu integrieren.

Einfachere Anwendungen, welche auf wenige Personen beschränkt sind, werden vermehrt durch diese selbst entwickelt werden. Komplexere Anwendungen, bei denen die Daten unternehmensweit zusammenhängen, oder auch umfangreichere technische Anwendungen werden nach wie vor den Einsatz von Spezialisten erfordern. Im produktionstechnischen Bereich bleibt das Problem der Verbindung des Computers mit der Anlagensteuerung. Hier können Standardisierungen, wie sie beispielsweise das bei General Motors in Entwicklung befindliche MAP (Manufacturing Automation Protocol) darstellt, Fortschritte bringen. Ganz allgemein wird die Bedeutung der Kommunikation zwischen verschiedenen Systemen wesentlich an Bedeutung gewinnen, denn der Nutzen der Datenverarbeitung wird noch wesentlich erhöht, wenn die für eine Aufgabe nötigen und vorhanden Informationen direkt zugänglich sind.

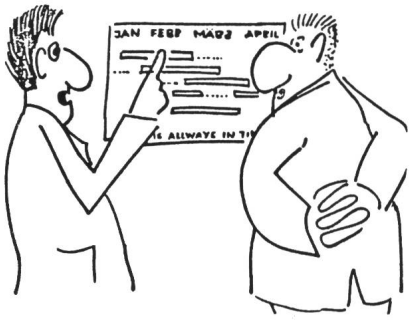

Fig. 6 Terminplan