Begriffliches

Organisation $\rightarrow$ beschreibt das geordnete arbeitsteilige Zusammenwirken mehrerer Komponenten eines Systems bei der Durchfuehrung zielgerichteter Handlungen

wesentlich sind $\rightarrow$ Disziplinen | Mechanismen des geordneten Zusammenwirkens

abstrakte Beschreibung durch
$\rightarrow$ Agenten | Aktoren (aktive Komponenten) $\rightsquigarrow$ fuehren Transaktionen | Operationen auf (Darstellungen von) Nachrichten | Dokumenten | Objekten einer realen | virtuellen Welt aus
$\rightarrow$ Kanaele (passive Komponenten) $\rightsquigarrow$ dienen als Traeger von Nachrichten | Objekten ... beim Transport zwischen Komponenten

System $\rightarrow$ ist Menge aktiver | passiver Komponenten, die zweckgebunden miteinander kooperieren | kommunizieren
$\rightarrow$ wird beschrieben durch
$\rightsquigarrow$ Eigenschaften der Komponenten und deren Wechselwirkungen
$\rightsquigarrow$ Festlegung von Systemgrenzen
$\rightsquigarrow$ Beziehungen zwischen Ein-/ Ausgabe

Rechnerorganisation betrifft

$\rightarrow$ Organisation von algorithmisch | deklarativ spezifizierten Berechnungs- | Entscheidungsvorgaengen | Prozessen
$\rightarrow$ Programmausfuehrendes System

Aktoren $\rightsquigarrow$ sind vom System unterstuetzte Operatoren
Kanaele $\rightsquigarrow$ sind Speicherbereiche | Laufzeitumgebungen zur Kommunikation von Operanden zwischen Operatoren

$\rightarrow$ Ablaufplanung | Ressourcenverwaltung fuer das programmausfuehrende System | Abwickelung von Kommunikationsvorgaengen | Verwaltung grosser Datenbestaende
$\rightarrow$ Betriebssystem

Aktoren $\rightsquigarrow$ sind primaere | sekundaere Scheduler fuer Prozesse | Transaktionen | ...
Kanaele $\rightsquigarrow$ sind diverse Datenstrukturen zur Darstellung von Prozess | Ressource | Systemzustaenden

Rechner(System)Architektur betrifft

$\rightarrow$ Gestaltung | Festlegung des Erscheinungsbildes | der Schnittstellen von Rechnersystemen gegenueber dem system-nahen Benutzer $\rightsquigarrow$ System(kern)programmierer | Implementierer von Compiler-backends | Code-Generatoren

Architektur
$\rightarrow$ des programmausfuehrenden Systems ist definiert durch
$\rightsquigarrow$ nicht-privilegierten Instruktionssatz | Adressierungsmodes | Elementardatentypen (Formate)
$\rightsquigarrow$ direkt zugaengliche Ressourcen (Funktionseinheiten | Register | Speicherbereiche ...)

$\rightarrow$ des Betriebssystems ist definiert durch
$\rightsquigarrow$ privilegierten Instruktionssatz (Prozessmanagement | i/o Vorgaenge | Prozess-Kommunikation)
$\rightsquigarrow$ Unterbrechungssystem | Prozesswechselmechanismen | ...

Architekturen definieren Systemfamilien mit identischer Funktionalitaet $\rightsquigarrow$ Maschinenprogramme werden auf allen Modellen der Familie auf gleiche Weise ausgefuehrt | produzieren exakt gleiche Ergebnisse

Essentials of system architecting

$\rightarrow$ bringing form to functionality | purpose
$\rightarrow$ fitting the interfaces (among components | to the user)
$\rightarrow$ reducing complexity
$\rightarrow$ balancing the needs
$\rightarrow$ compromizing among the extremes

$\rightarrow$ consistency (absence of contradictory behavior)
$\rightarrow$ no unnecessary functions (propriety)
$\rightarrow$ no functional reduncies in different diguises (parsimony)
$\rightarrow$ transparency
$\rightarrow$ generality and completeness of functions

$\rightarrow$ aiming for optimality $\rightsquigarrow$ a waste of time

$\rightarrow$ primary objective of architecting
    $\rightarrow$ KISS $\rightsquigarrow$ keep it simple, stupid !!!!

$\rightarrow$ well designed systems with good chances of long-term survival are the product of only a few !!!!

Computer Engineering

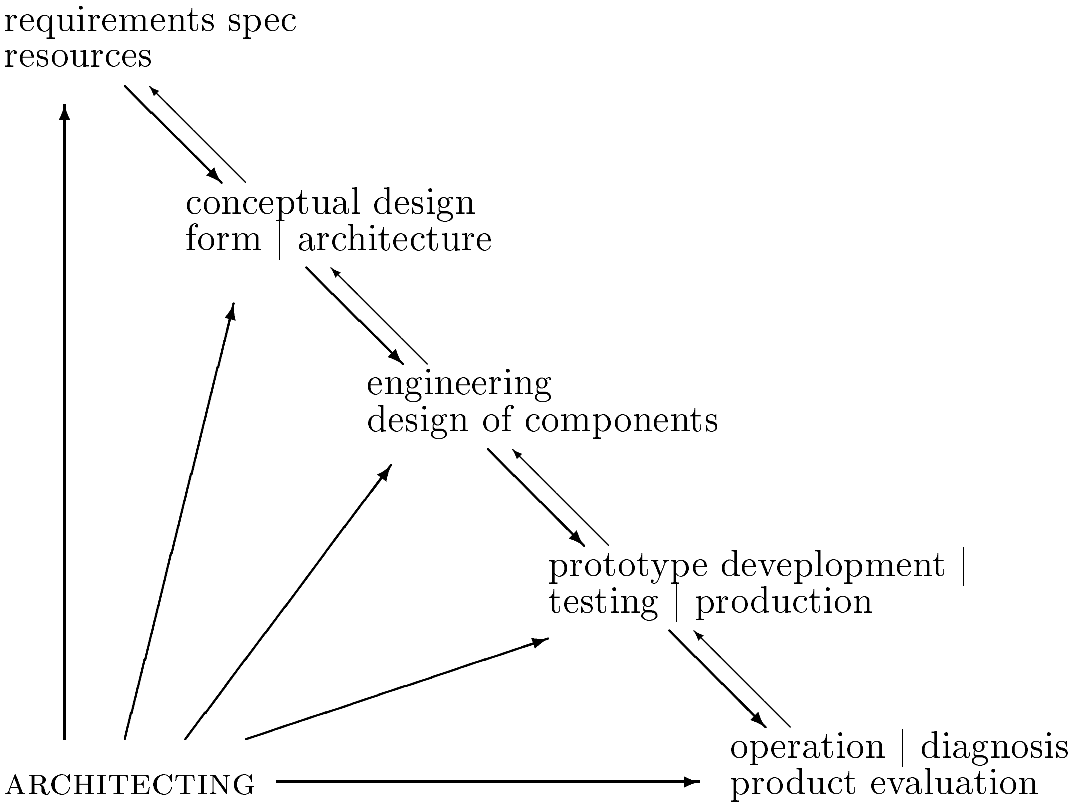
$\rightarrow$ technische Realisierung von individuellen Modellen einer Architektur | Systemfamilie

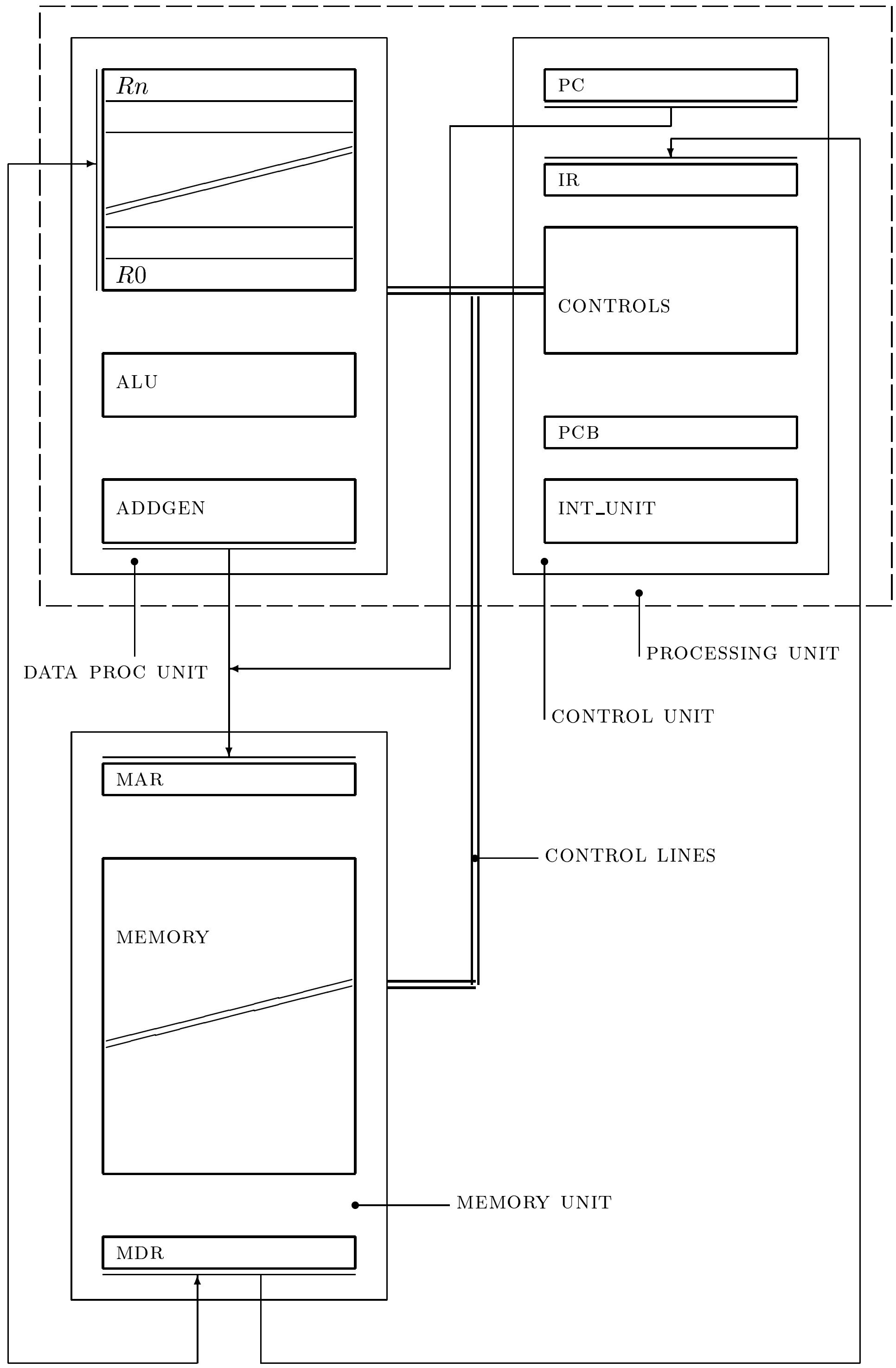
Wesentliche Gesichtspunkte sind
$\leadsto$ technologische Gegebenheiten
$\leadsto$ Produktions | Vertriebs | Wartungskosten
$\leadsto$ Leistungs | Kapazitaets | Zuverlaessigkeitsanforderungen
$\leadsto$ Details der Soft | Firm | Hardware Implementierung (wird ggf. der Architektur zugerechnet | nicht genau zu trennen)

Wichtig :

$\rightarrow$ das Architecting beeinflusst alle Phasen der Systementwicklung | Fertigung | Nutzung
$\rightarrow$ idR muessen die einzelnen Phasen der Systementwicklung mehrfach durchlaufen werden, da sie sich wechselseitig beeinflussen

$\leadsto$ Wasserfall Modell


requirements spec
resources
conceptual design
form | architecture
engineering
design of components
prototype deveplopment |
testing | production
operation | diagnosis
product evaluation
ARCHITECTING



Rn
$R0$
ALU
ADDGEN
PC
IR
CONTROLS
PCB
INT_UNIT
DATA PROC UNIT
PROCESSING UNIT
CONTROL UNIT
MAR
MEMORY
CONTROL LINES
MEMORY UNIT
MDR