Grundlagen der Echtzeitplanung

1. Modellbildung

2. Planungsverfahren (Scheduling)
   - Planen durch Suchen
   - Planen nach Fristen
   - Planen nach Spielräumen
   - Planen nach monotonen Raten

3. Kritik der Planungsverfahren
4. **Adaptive Planungsverfahren**
5. **Garantien**

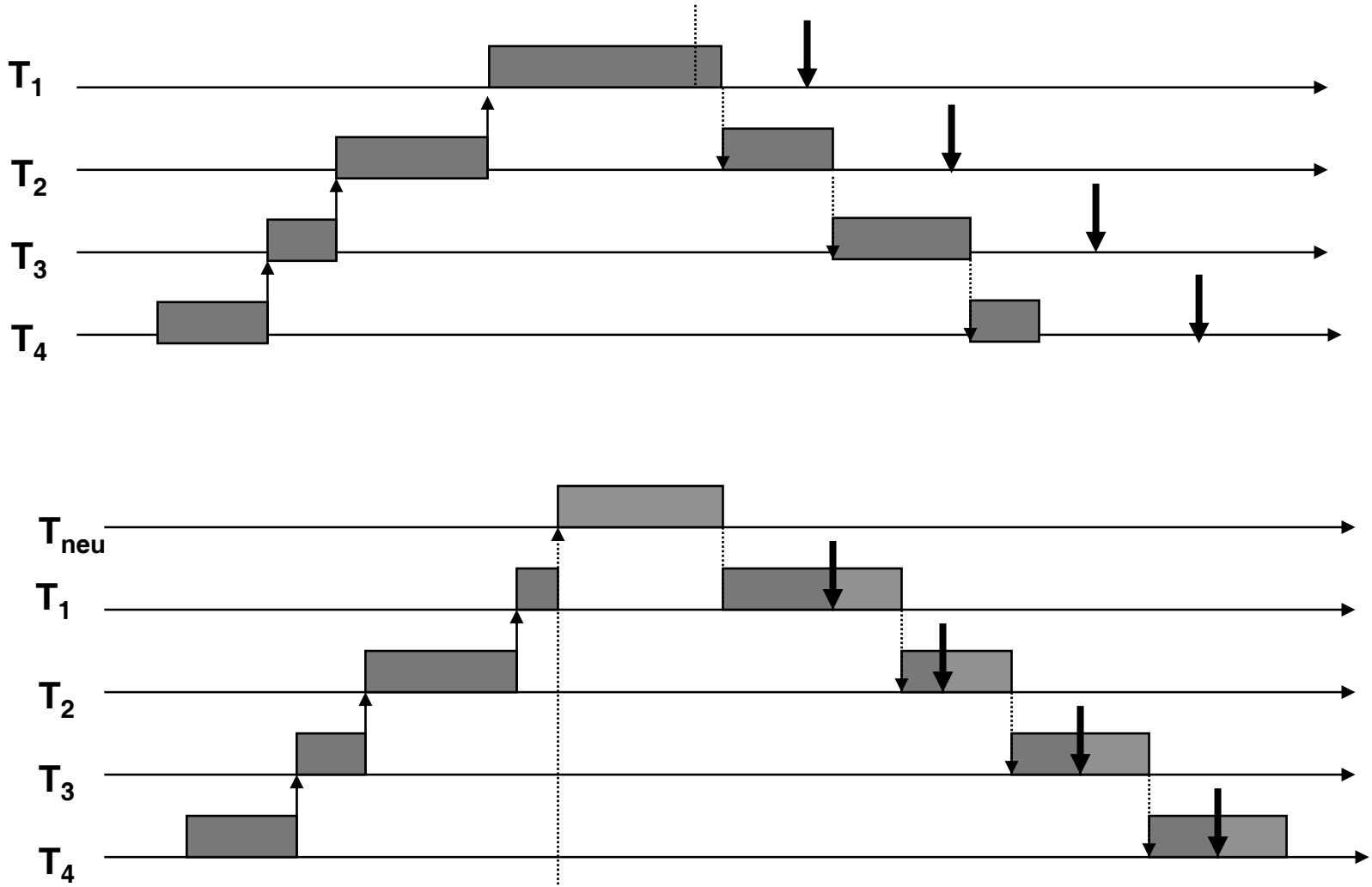
Probleme:

1. **Behandlung von Überlast**
2. **Integration von Tasks unterschiedlicher Kritikalität**
3. **Integration periodischer und aperiodischer und sporadischer Tasks**

Überlast entsteht:

1. **Durch Überschreiten der WCET**
2. **Durch neu hinzukommende Tasks**

Der Domino-Effekt durch neu hinzukommende Tasks



1. **Überschreitung der WCET.**
   Kommt in „harten Echtzeitsystemen“ nicht vor, d.h. diese Systeme bauen auf eine Fehlervermeidungsstrategie.
   Einige Verfahren ermitteln die WCET nur für einen Teil der Gesamtaufgabe und nutzen möglicherweise verfügbare Zeit zur Qualitätsverbesserung des Ergebnisses.
   - **imprecise computations**
   - **TAFT (Time Aware Fault-Tolerant) Scheduling**

   Andere Verfahren nutzen zusätzlich Informationen über die Tasks wie z.B. die Wichtigkeit, den Spielraum etc. zur Entscheidung aus.

2. **Neue Task kommt hinzu.**
   Was benötigt wird, ist ein Entscheidungskriterium, ob eine neue Task zugelassen werden kann.

   - **Verfahren mit Akzeptanztest**
   - **Verfahren mit Akzeptanztest und dynamischer Ermittlung des verfügbaren Spielraums**

Bisher:

- **Unterlast, d.h. $U \leq 1$**
- **keine dynamische Taskaktivierung, d.h. alle Tasks sind bereits vorhanden.**

Neues Problem:

Dynamische Taskaktivierung

Überlastsituation

Einige Tasks können nicht rechtzeitig ausgeführt werden

Rangfolge unter den Tasks ?

Imprecise Computations

Jane Liu, Kwei-Jay Lin, Wei-Kuan Shih, Albert Chuang-shi Yu(Uni Illinois @ Urbana-Champaign)
Jen-Yao Chung (IBM), Wei Zhao (Texas A&M Uni)
IEEE Computer 1991

Imprecise Computation ist ein Ansatz, der transiente Überlast behandelt, wie sie z.B. durch variable Ausführungszeiten hervorgerufen werden kann.

Assumption: Monotone, time-critical computations

Monotone means that the quality of the result is a function of time and that the quality of intermediate results do not decrease as the task executes longer.

Approaches:

Milestones:

Application dependent milestones define certain quality levels for the imprecise result.

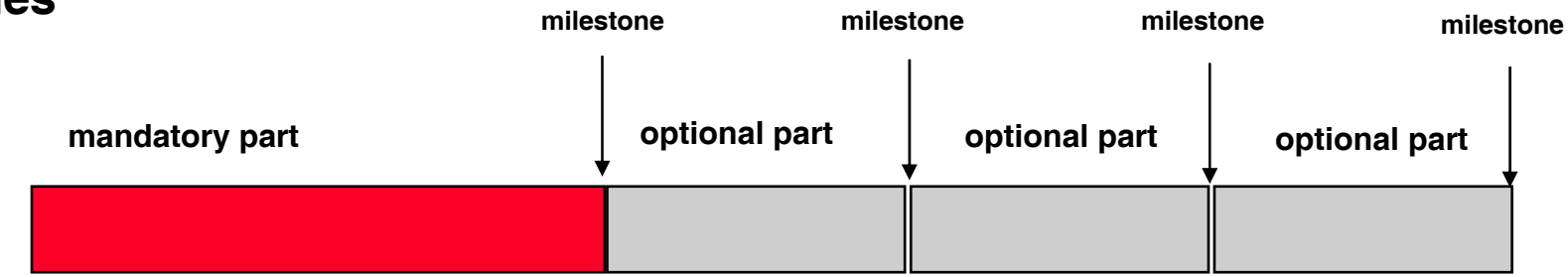
Sieve Functions:

Computation Steps or Subroutines are skipped.

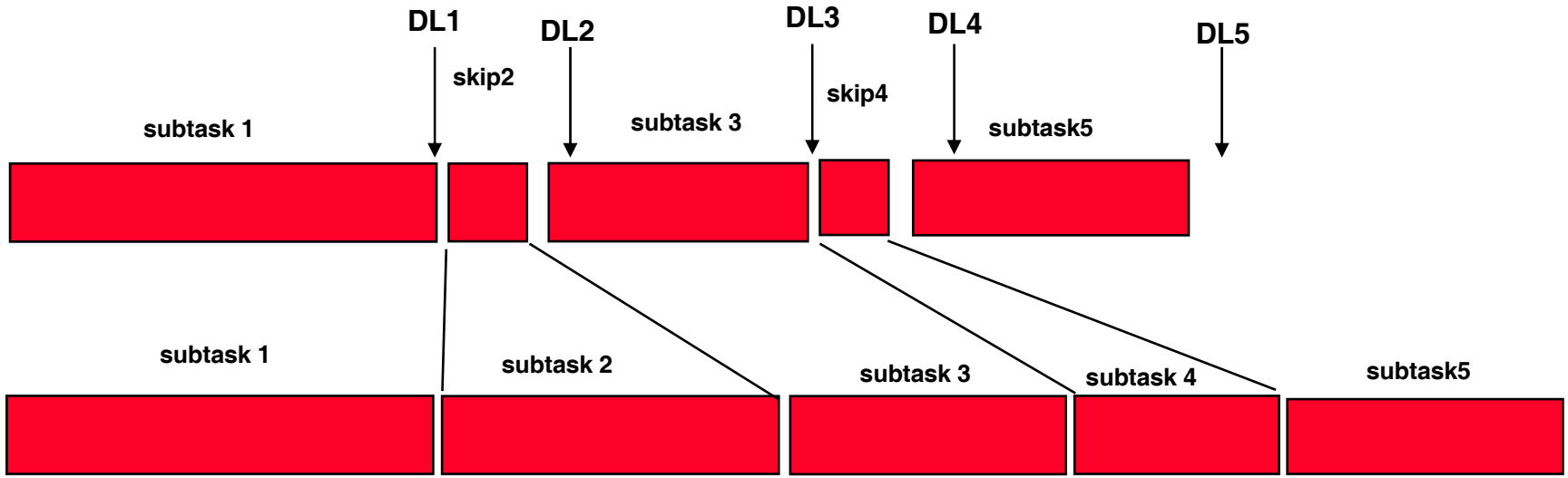
Multiple versions:

For each computation a primary and a secondary version of an algorithm exist. The primary produces better results while the secondary executes faster. If the system detects transient overload, it switches to the secondary.

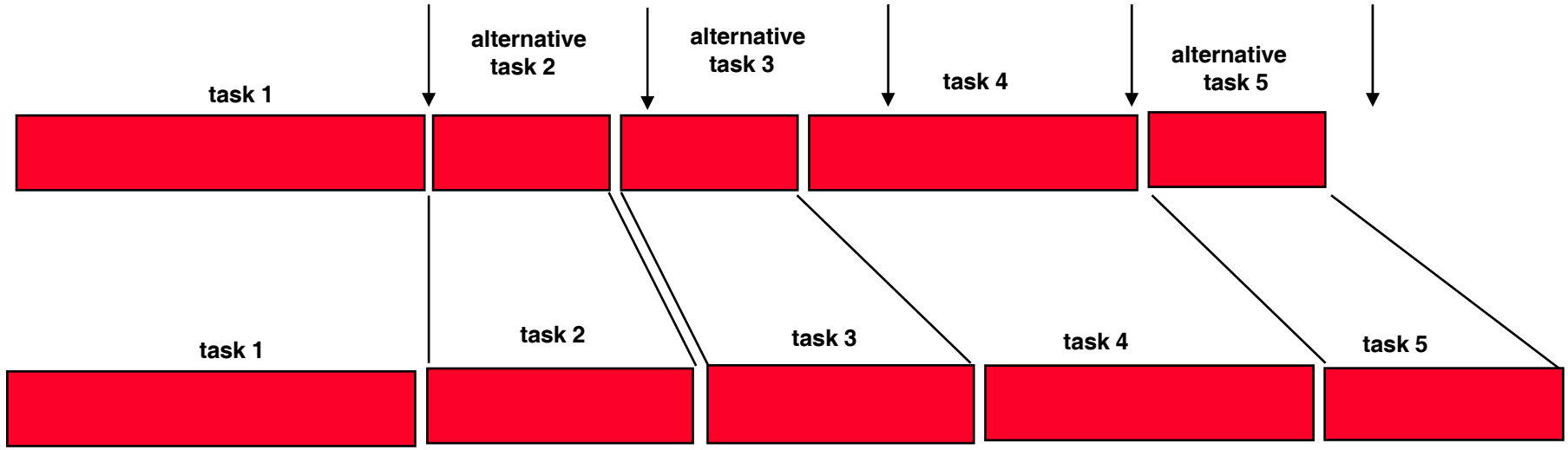
Milestones



Sieve



Multiple versions



Method:	Overhead:
Milestone:	Defining and recording intermediate results
Sieve:	Determining overload situation, higher scheduling overhead, subtask have the all-or-nothing (0/1) property
Multiple Versions:	Defining multiple versions, storing multiple versions, higher scheduling overhead

Dynamisches Scheduling und Gararantie

TPS : Task Pair Scheduling
TAFT : Time Aware FT Scheduling

M.Gergeleit, J. Kaiser, H. Streich
DIRECT- Towards a Distributed Object-Oriented Real-Time System
Workshop on Concurrent Object-Oriented Systems, Dallas TX, October 1994

Nett, E., M. Gergeleit, M. Mock.
An Adaptive Approach to Object-
Oriented Real-Time Computing, Proc. ISORC'98, Kyoto, Japan,
Apr. 1998.

TPS verbindet die Idee der on-line-Garantien mit dem Konzept des Exception Handlings.

Durch On-line-Garantien wird Vorhersagbarkeit im System unterstützt.

Die Verbindung mit Exception Handling ist ein Ansatz zur Lösung:

1. des Problems der nicht bestimmbaren WCET in komplexen Anwendungen.
2. des Problems harte und weiche Zeitbedingungen mit einem Konzept zu behandeln.
3. des Problems Ausnahmesituationen, wie Fehler oder Zeitüberschreitungen zu behandeln.

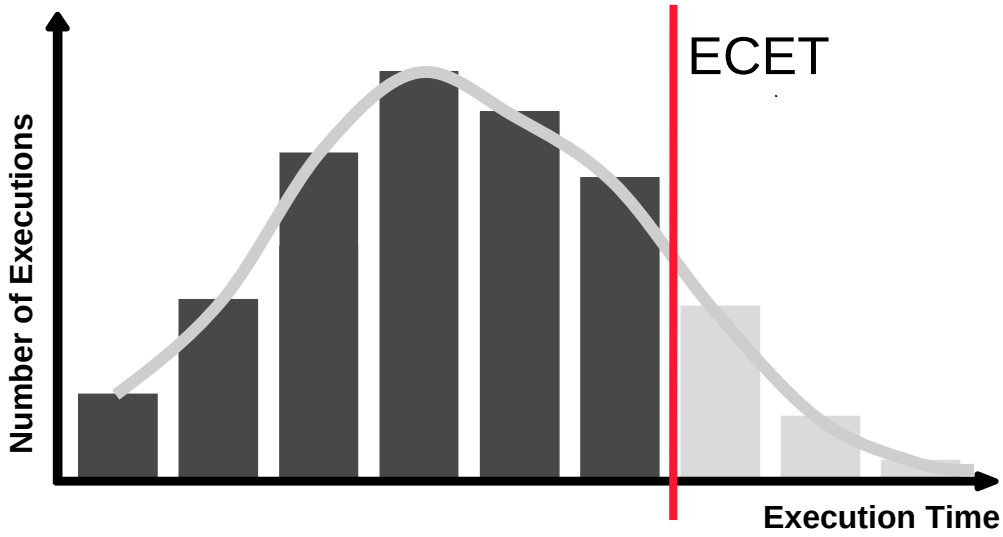
Problems with WCET's:

- dependent on hardware architecture,OS, compiler, PL $\Rightarrow$ difficult to predict
- many features serve to improve average case behavior, NOT worst case behavior

  Examples:
  - caches, pipelining, virtual memory
  - interrupt handling, preemptions
  - optimizing compilers
  - recursions

- **Even more difficult if depending on the environment (embedded systems)**

Expected Case Execution Time



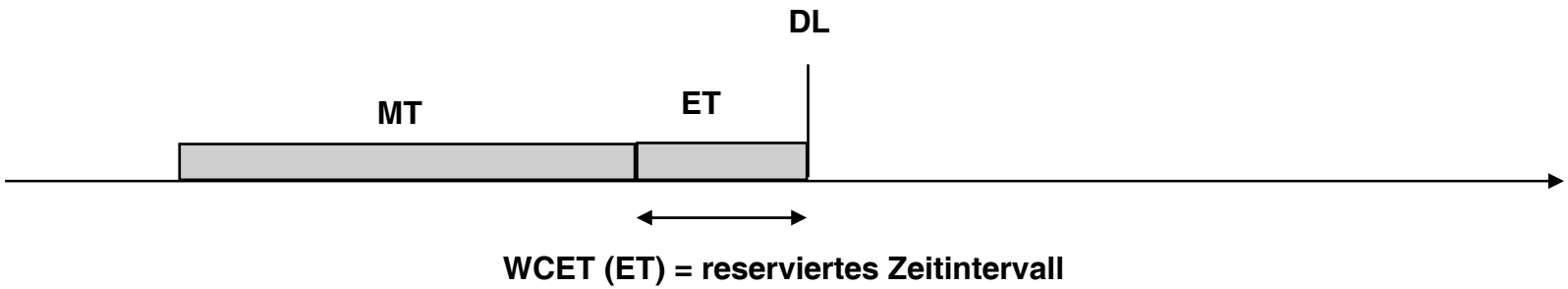
ECET can be obtained by evaluating the recent executions of the task.

- number of tasks below a certain execution time bound.
- average execution time.

Task-Pair-Scheduling

```
guarantee (TP, DL)
else  /* no guarantee */
```

```
taskpair TP (deadline)
{  try_within (OCET,deadline)
        MainTask;    /* soft real-time */
   except
        ExceptionTask;  /* critical */
}
```



Goals of TAFT (Time - Aware Fault - Tolerant) Scheduling

- **No Handling of tasks with unknown or too pessimistic WCETs**
  - **Introduction of Expected Case Execution Time (ECET)**
- **Still with Timing Guarantees**
  - **Scheduled exception handling** before **the deadline**
- **Fault-Tolerance with respect to timing errors**
  - **Graceful degradation in overload situations**
  - **Tradeoff between functionality and timing**

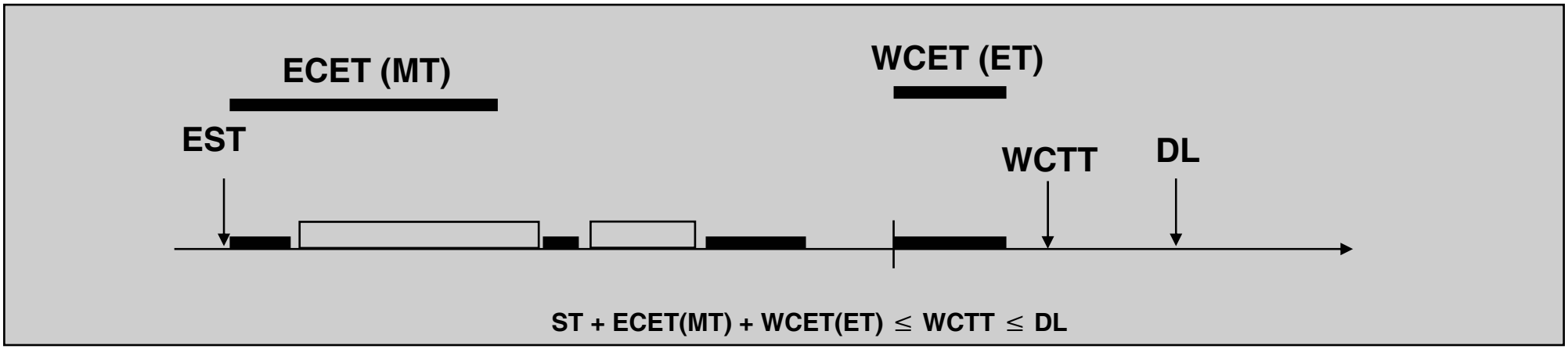
TPS/TAFT-Scheduling

Guarantee:

Task ends in a defined state before the deadline.

Taskdescriptor:

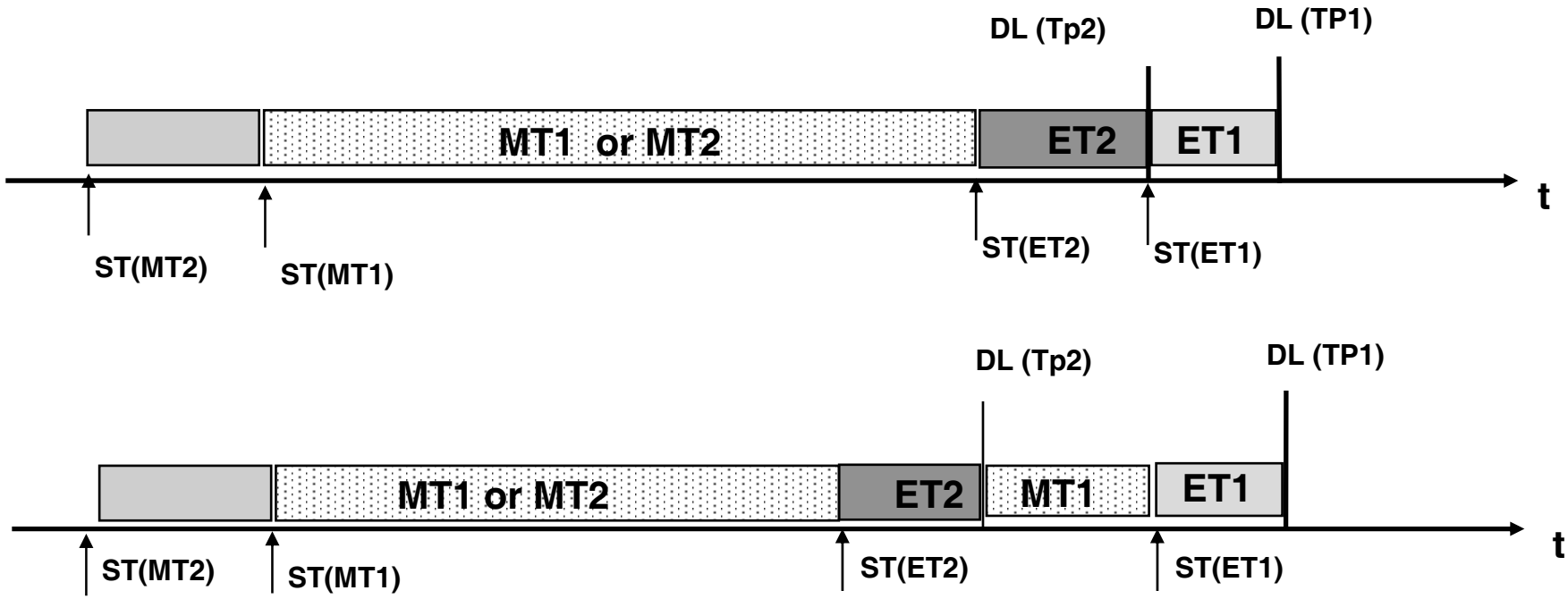
```
int   importance;

/*  time specifications  (assumptions) */
int          DL;                  /* deadline */
int          WCET;                /* worst case execution time
                                     optional for MT*/
int          EST;                 /* earliest start time (ready time) */
int          ECET;                /* expected case execution time
                                     not relevant for ET */

/* Parameters computed by the scheduler */
int          ST;                  /* (actual) start time */
int          WCTT;                /* worst case termination time */
```



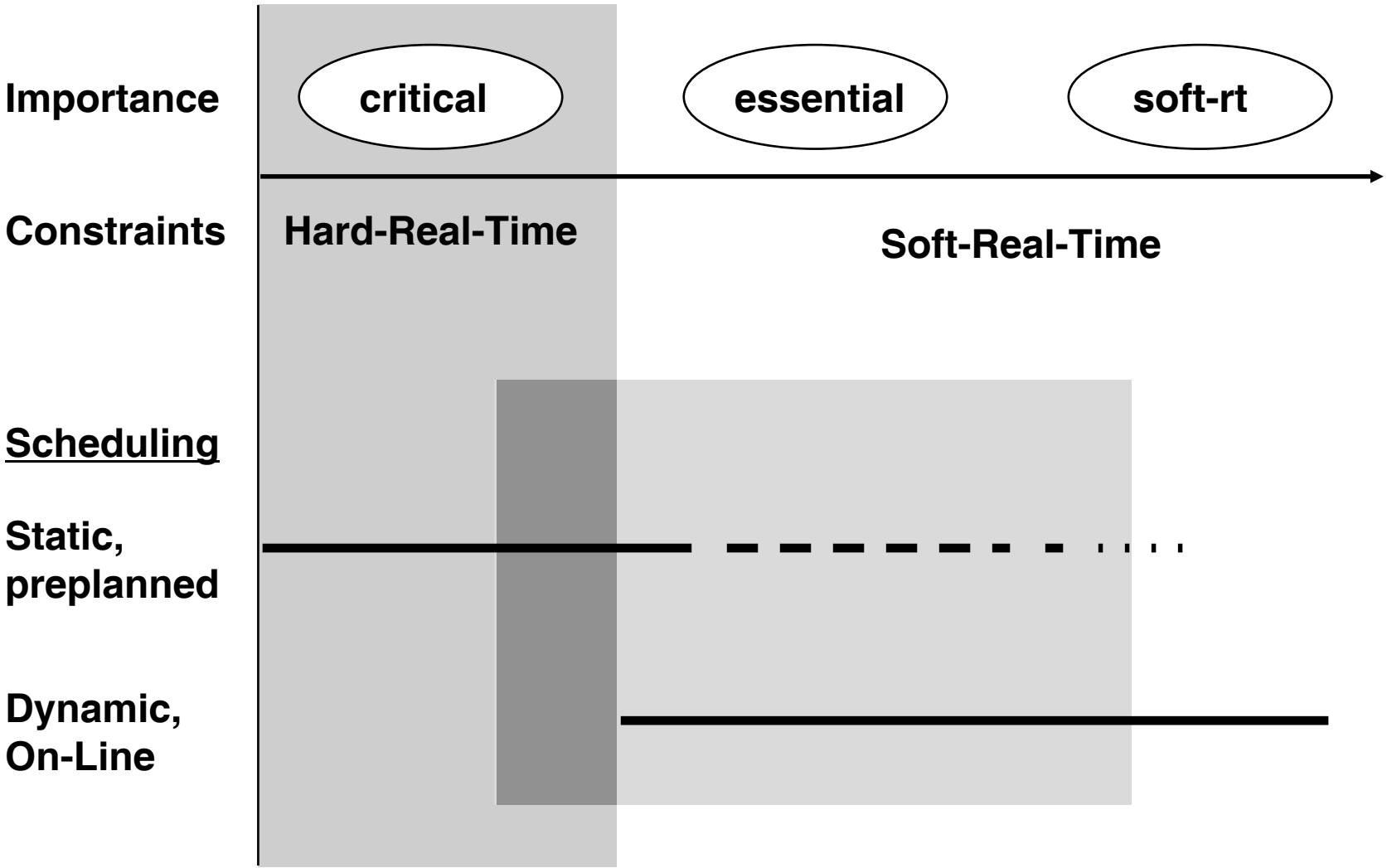
Basic-Algorithm

```
task TP1 (deadline)
{  try_within (ECET,deadline)
        MT1;    /* soft real-time */
   except
        ET1;  /* essential */
}
```

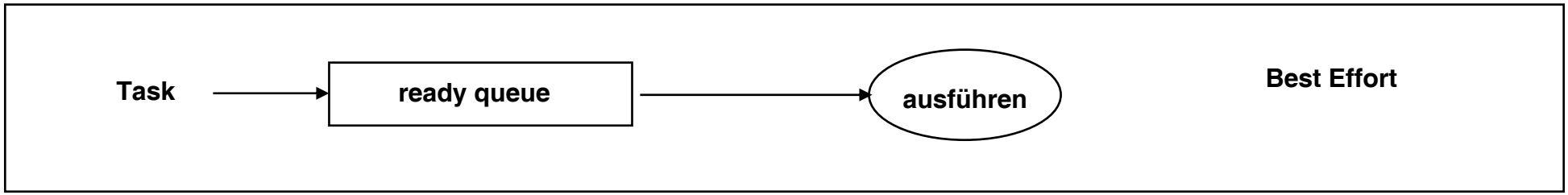
```
task TP2 (deadline)
{  try_within (ECET,deadline)
        MT2;    /* soft real-time */
   except
        ET2;  /* essential */
}
```

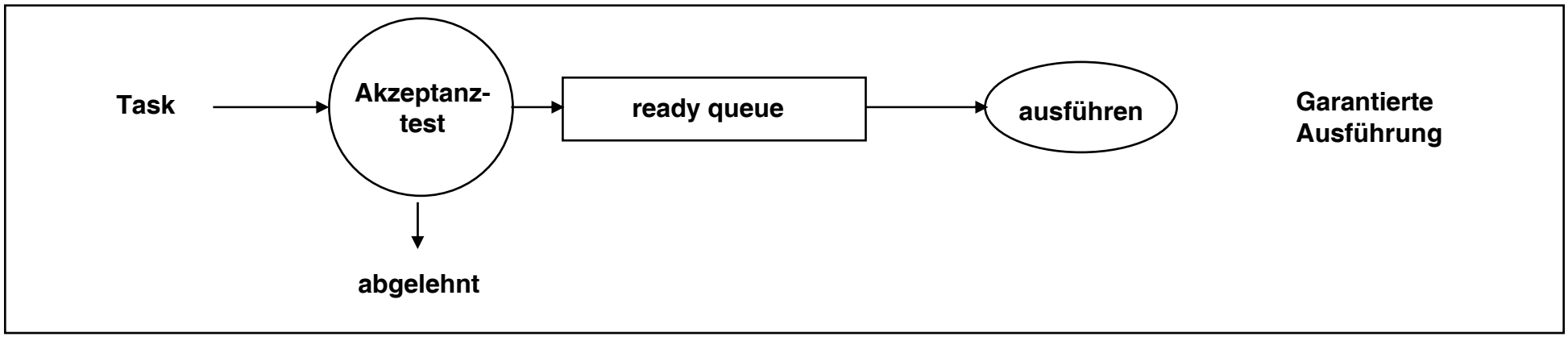


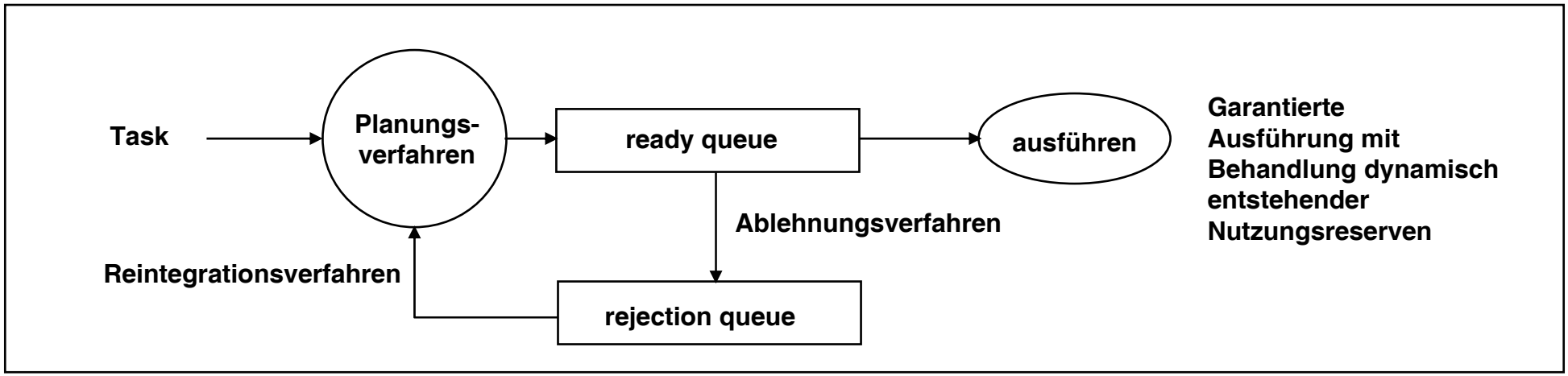
Classifying Tasks



Garantien, wenn neue Tasks hinzukommen können







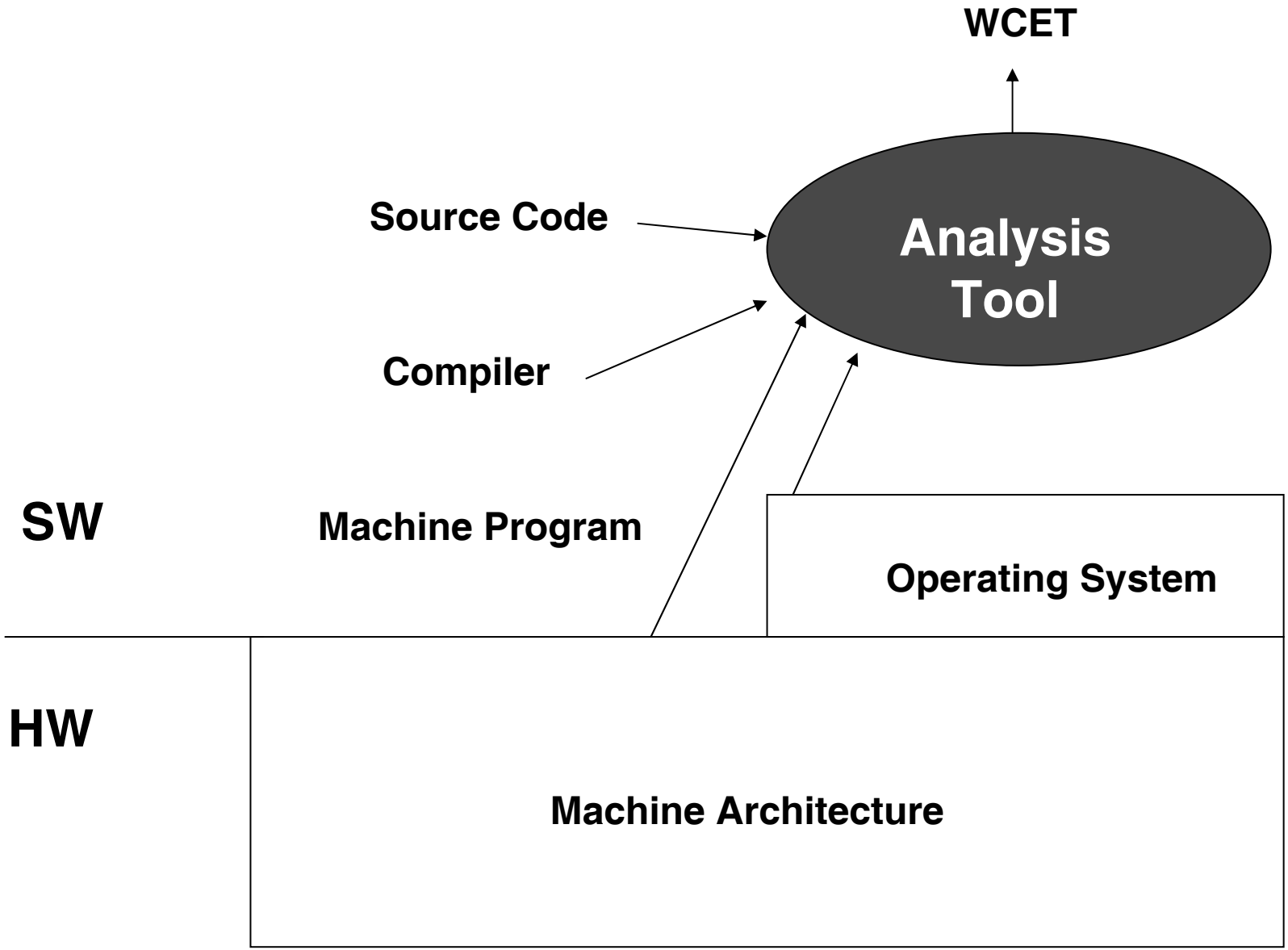
Abschätzung der Ausführungszeiten

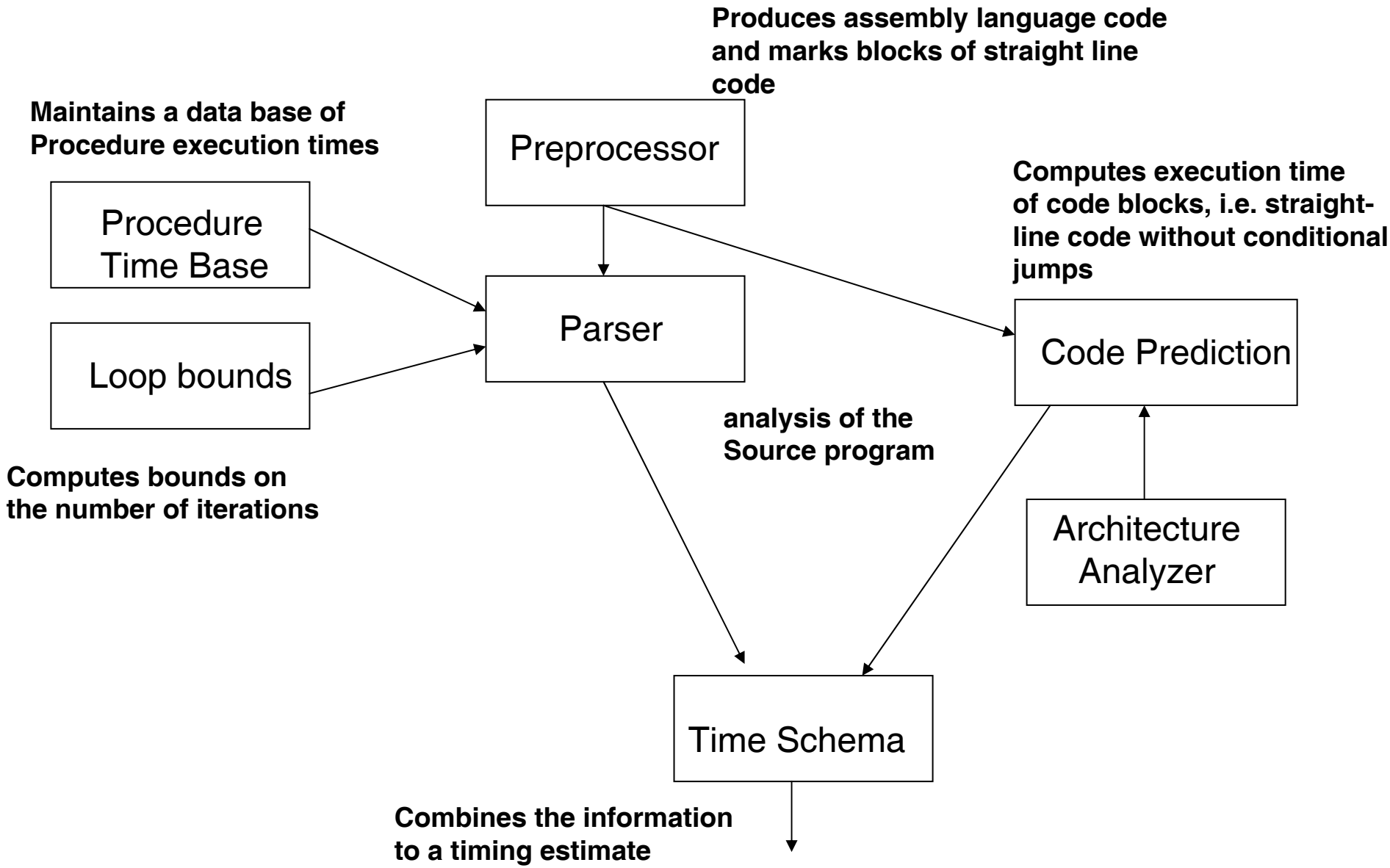
Probleme

Anwendung	Kontrollschleifen Anzahl der Eingangsparameter Qualität der Eingangsparameter Interner Zustand
Betriebssystem	Virtueller Speicher, Plattenzugriff (Reordering), Synchronisation, Systemaufrufe, Resourcenverwaltung (shared res.)
Hardware	Pipeline, Caches, Dynamic RAM, Network

Die Worst Case Execution Time kann nicht experimentell bestimmt werden !!!!

- **Die Menge der möglichen Eingabewerte ist zu groß,**
- **Das Programm läuft nicht isoliert ab.**
- **Durch häufige Testläufe kann nur die Average Case Execution Time und die Abweichung bestimmt werden.**


WCET
Source Code
Analysis Tool
Compiler
SW
Machine Program
Operating System
HW
Machine Architecture



C.Y. Park, A. Shaw: „Experiments with a Program Timing Tool Based on Source-Level Timing Schema“, IEEE Comp., 24(5), 1991

Monitoring

Überwachung (insbesondere) zeitlicher Spezifikationen in verteilten dynamischen Steuer-und Kontrollsystemen

- **Überwachung während der Missionszeit des Systems**
  - **Reaktion des Systems auch bei internen Fehlern**
  - **Initialisierung einer adaptiven Reaktion z.B. durch**
    - **> Aktivierung alternativer Ressourcen**
    - **> Aktivierung alternativer Algorithmen**
    - **> Rescheduling vorhandener Ressourcen**

- **Beitrag zur Überlebensfähigkeit des Systems**

Wie erhält man die notwendige Information aus dem System?

Monitoring und Adaptivität