A Basis for Performance Property Prediction of Ubiquitous Self-Adapting Systems

Gunnar Brataas, Jacqueline Floch
SINTEF ICT, Norway

Romain Rouvoy
University of Oslo, Norway

Pyrros Bratskas, George Papadopoulos
University of Cypros

Engineering of Software Services for Pervasive Environments (ESSPE '07), Dubrovnik

Self-Adapting Applications for Mobile Users in Ubiquitous Computing Environments
MUSIC, Integrated Project, 6th FP
www.ist-music.eu

4 September 2007

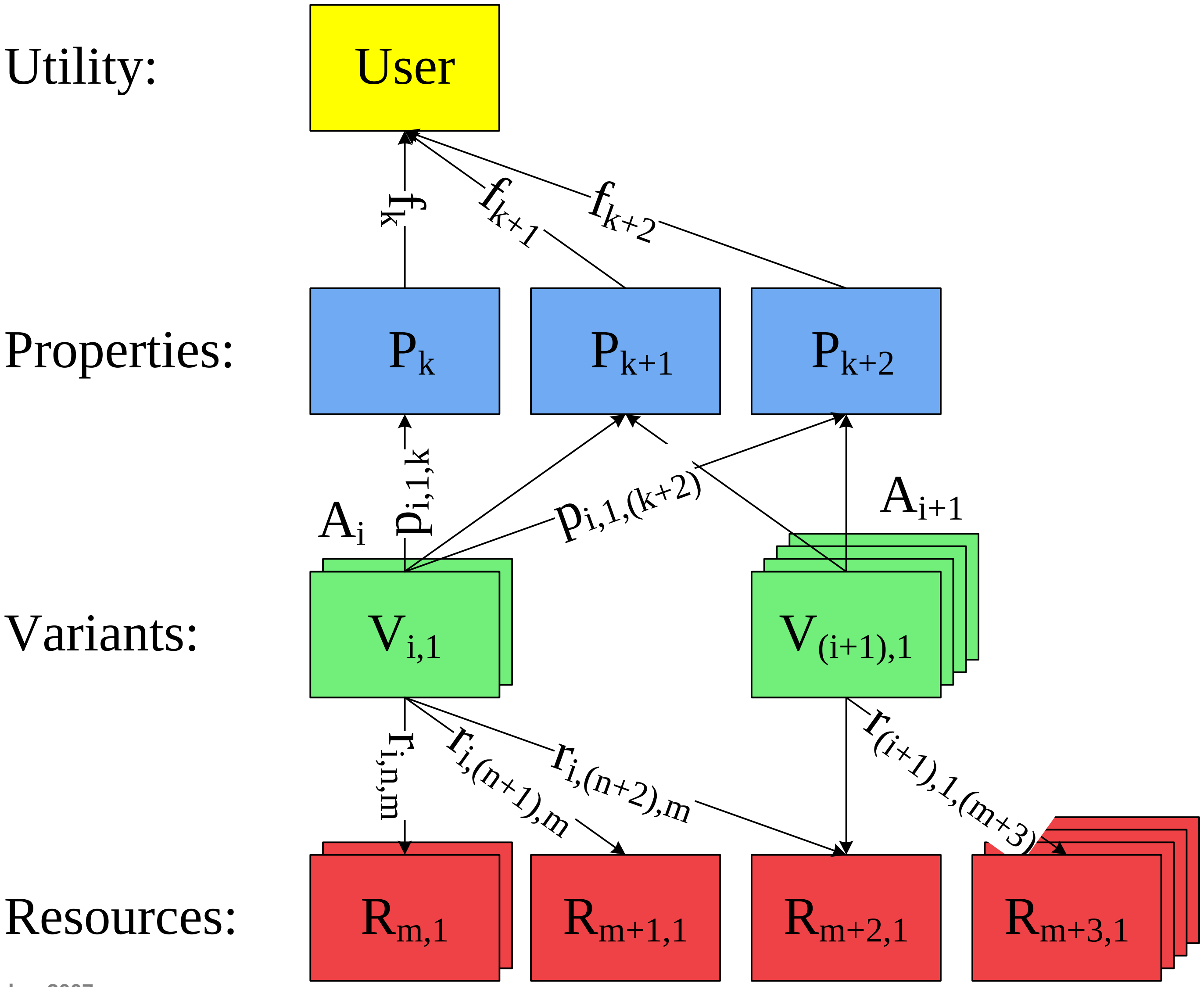
The “Essence” of the MUSIC Project

- Component-based mobile system
- Each component has component variants
- Each component variant has property predictors
  - Specified by developers
- All permutations of all component variants gives the application variants for each application
  - Depending on context, select “best” application variant for all applications

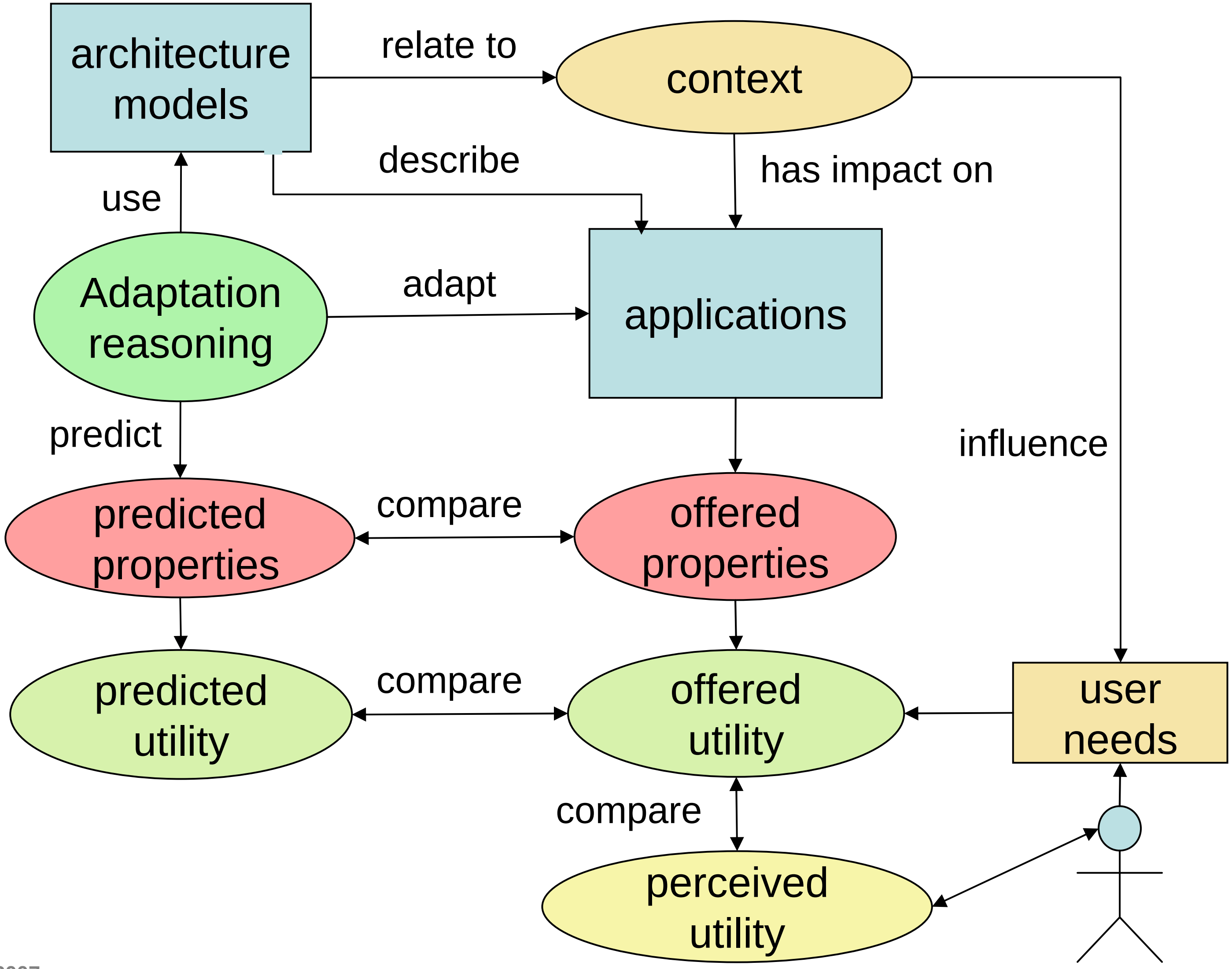
Objectives

- Assist component developers in component performance property prediction
  - Average developer does not know much about performance modelling and measurement
- Develop formal basis for performance property prediction of mobile systems
  - Components are integrated on the fly
  - Assumption: Better to simplify a rigorous framework than to work with an ad hoc approach

Basic Concepts



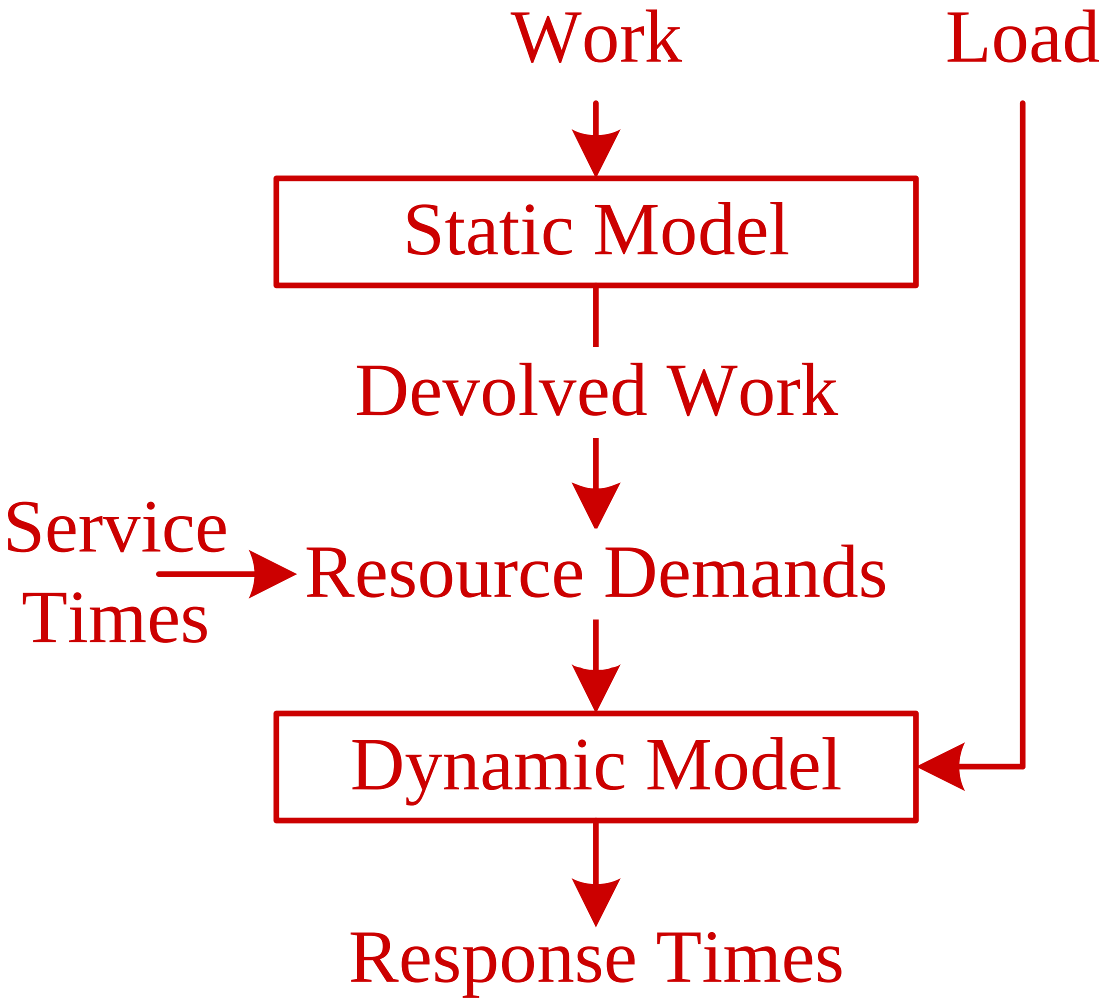
Overall Framework



Component-based performance engineering

- Component-based paradigm
  - Static performance model for SW components
    - No contention (or queuing) for software resources
    - Hughes:88, Vetland:93, Brataas:96
  - Dynamic performance model for HW resources
    - Classical queueing network models, with contention
- Competing paradigms
  - Software Performance Engineering (Smith:90,02)
    - Static and dynamic, weak on hierarchies and components?
  - Layered queueing networks (Rolia:96,Woodside et al.)
    - Purely dynamic, more complex
  - Interesting to explore them too

Combining Static with Dynamic Models



Work

Load

Static Model

Devolved Work

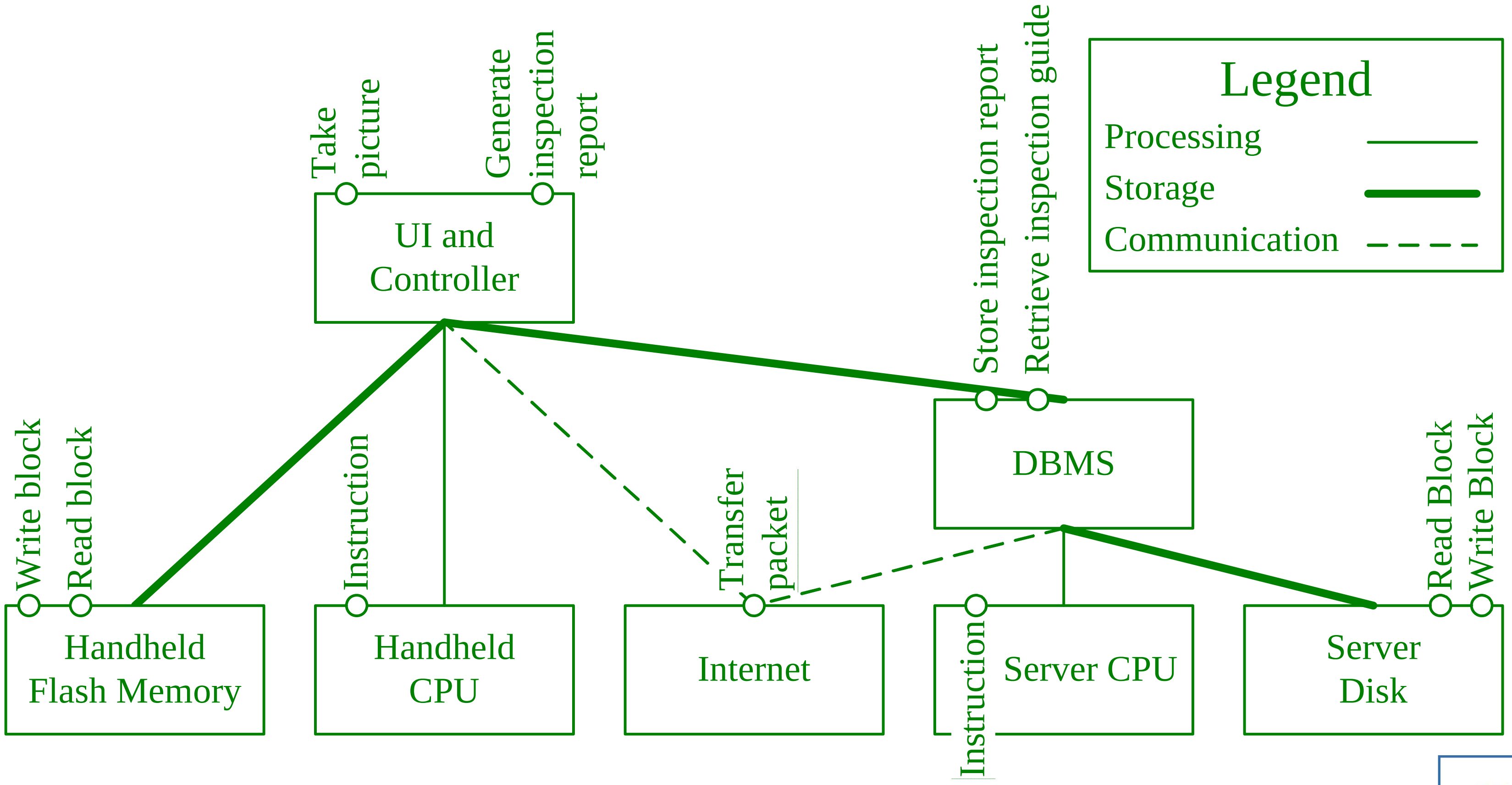
Service Times

Resource Demands

Dynamic Model

Response Times

Static Model of Service Technician App.



Complexity Specification Matrices

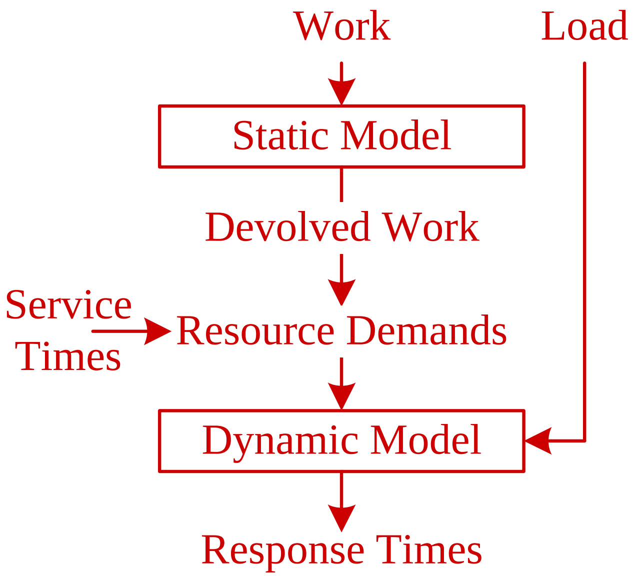
$$C^{UI\_\&\_Cntrl}_{Flash\_mem} = \begin{array}{lcc} & RB & WB \\ Take\_picture & 0 & 10 \\ Generate\_report & 100 & 1000 \end{array}$$

$$C^{UI\_\&\_Cntrl}_{CPU} = \begin{array}{lc} & Instr. \\ Take\_picture & 10^{7} \\ Generate\_report & 10^{8} \end{array}$$

Calculating Resource Demands

$$C^{UI\_\&\_Cntrl}_{Flash\_mem} = \begin{array}{ccc} & RB & WB \\ Take\_picture & 0 & 10 \\ Generate\_report & 100 & 1000 \end{array}$$

- Each RB requires 0.02 s and each WB 0.05 s
- Resource demands (D):



$$D_{T\_p,mem} = 10 \cdot 0.05s = 0.5s$$

$$\begin{aligned} D_{G\_r,mem} &= 100 \cdot 0.02s + 1000 \cdot 0.05 \\ &= 2s + 50s = 52s \end{aligned}$$

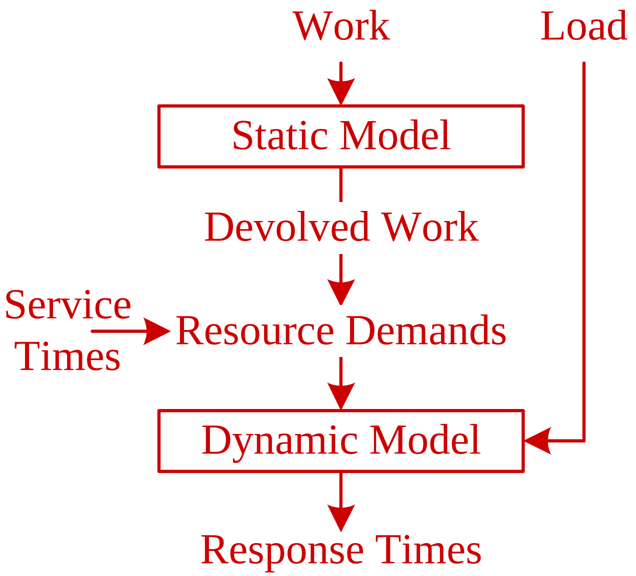
Calculating Response Times

- Calculating utilisation (U),
  1 G_r and 20 T_p per
  20 minutes:

$$
\begin{aligned}
U &= \frac{L_{T\_p}\,(D_{T\_p,CPU} + D_{T\_p,mem}) + L_{G\_r}\,(D_{G\_r,CPU} + D_{G\_r,mem})}{10 \cdot 60\,s} \\
&= \frac{20\,(1\,s + 0.5\,s) + 1(10\,s + 52\,s)}{10 \cdot 60\,s} = 0.153
\end{aligned}
$$

- Calculating response time
  for Generate report (R):

$$
R = \frac{D}{1-U} = \frac{10\,s + 52\,s}{1-0.153} = 73.2\,s
$$



Open Research Questions

- Emerging directly from the work presented
  - In MUSIC coarse grained architectural model: services
    - Fine-grained: individual operations
  - Variability requires new CSMs?
  - Validation: Case studies using MUSIC pilot applications
  - Strike a good balance between measurement cost and prediction accuracy: practical experience needed
- For broader research community
  - CBPE still not normal practice: costly
    - Standardised test beds needed
    - CSM repositories: use existing measurements
    - Client part of mobile systems simpler than stationary systems?
  - Memory
    - Memory consumption of each component itself
    - Memory constraints in primary memory
  - Extent to model energy consumption
    - Energy consumption non-linear with CPU frequency