

Task-Oriented Systems for Interaction with Ubiquitous Environments

Chuong C. Vo, Torab Torabi , & Seng W. Loke
Department of Computer Science & Computer Engineering
La Trobe University, Australia

Presenter: Chuong C. Vo

Outlines

- Introduction
 - Ubiquitous environments and smart spaces
- Usability problems with smart spaces
- Approach: Task-Oriented Systems (TASKOS)
 - Concepts & System Architecture
- Implementation
- Related work
- Conclusion & future work

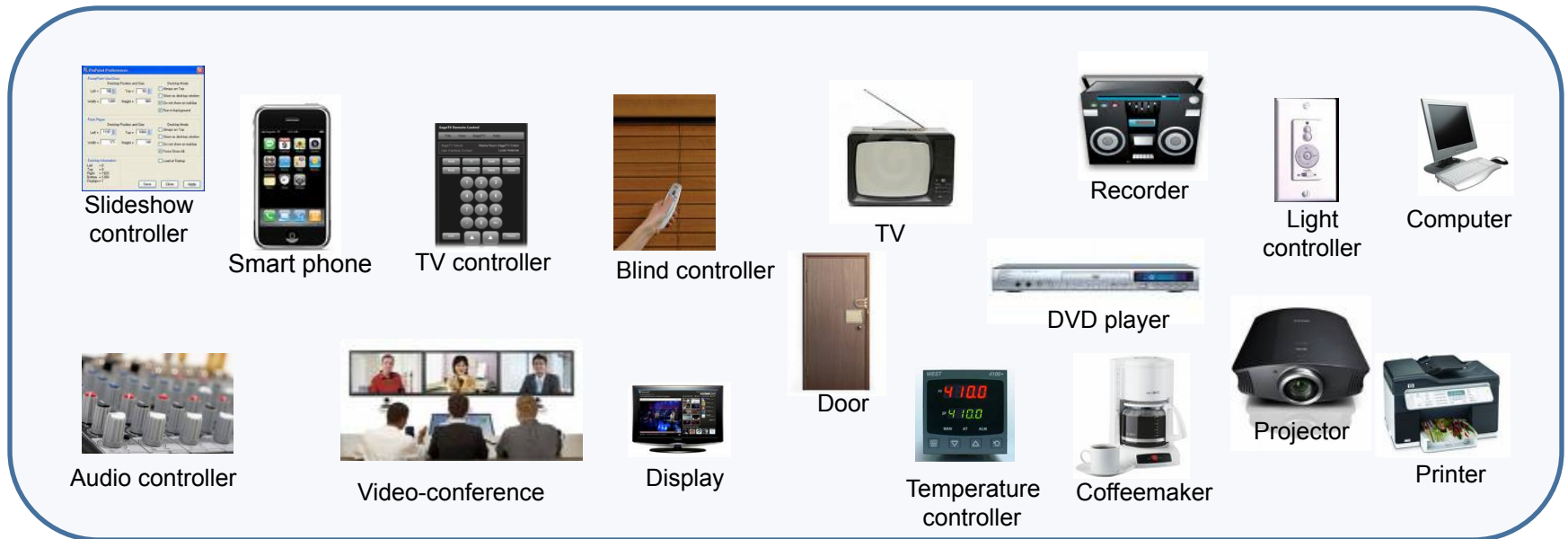
Introduction

- The third era of computing (post-PC)



- Pervasive (everywhere, every time), computational elements blend into “the fabric of everyday life”.
- Your rooms, offices, homes are computers.

Smart spaces



- The aims:
 - Support our activities, complement our skills, add to our pleasure, convenience, accomplishments
- [Norman 2007].

Usability problems with smart spaces

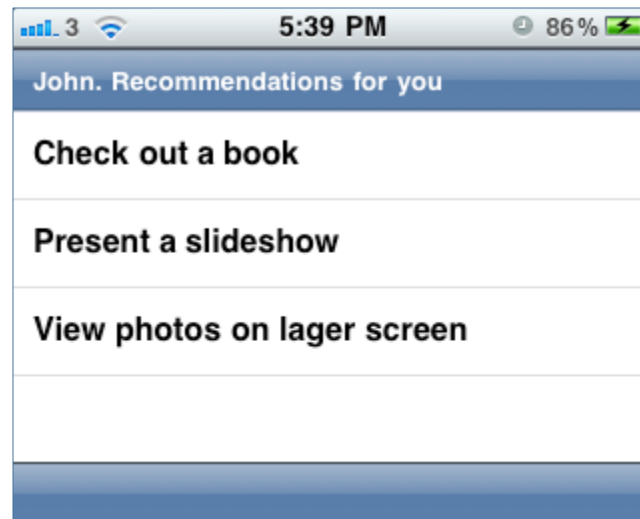
- **Complexity of use:**
 - Variety of devices, remote controllers, user interfaces (UI)
 - Too many buttons and menus on UI
 - Inconsistency of user interfaces
 - Users need to understand how to map devices' functions to their tasks
 - Inconsistency of task executions (task procedures)
- **Invisibility of features**
 - Technologies blend into environments
 - Frequently adding/removing devices & services to/from spaces
- **Overload of features**
 - One device provides tens of features; a combination of devices provides hundreds of features.

Approach: Task-Oriented Systems

- Our approach is based on task-driven computing [Wang et al. 2000]:
 - A *task* is a user's *goal* or *objective* [Loke 2009].
 - Users interact with a smart space in terms of *tasks* instead of *applications/devices' features*.
 - Users focus on their tasks at hand rather than on the means for achieving those tasks [Masuoka2003].

Features of TASKOS

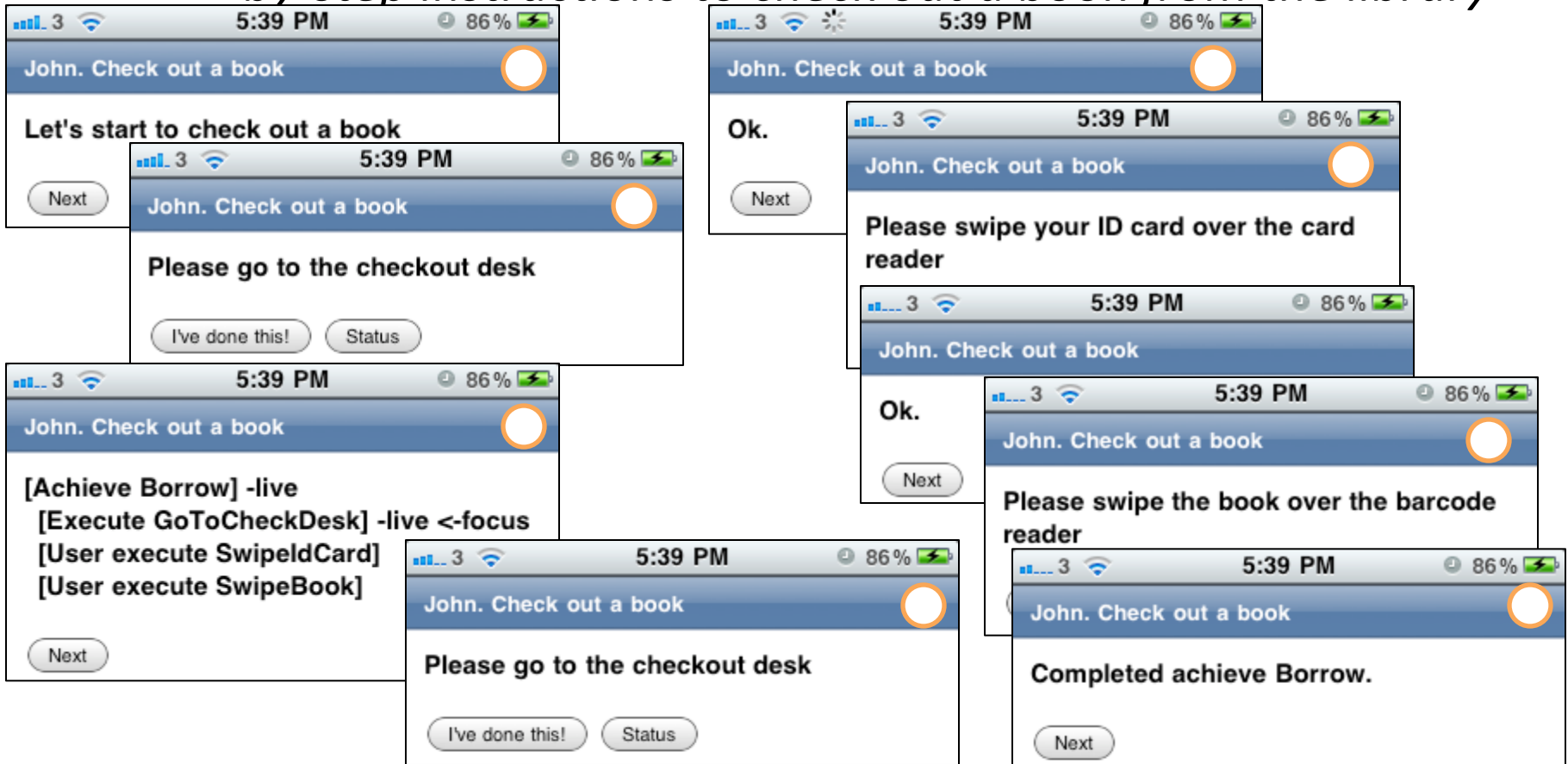
1. Recommend users for relevant and possible tasks based on context-awareness [Vo *et al.* 2009]
 - *It's 7p.m., it's raining, and you're walking in the centre of Melbourne. TASKOS suggests **Dinner?**, **Taxi?**, **Bus?***
 - *You're at the library, TASKOS would suggest*



Features of TASKOS (cont.)

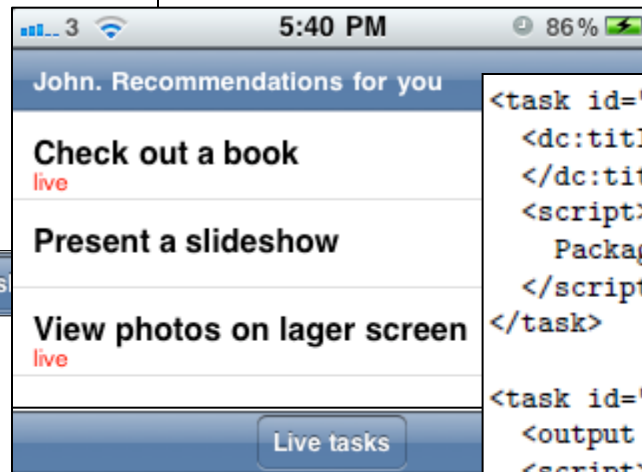
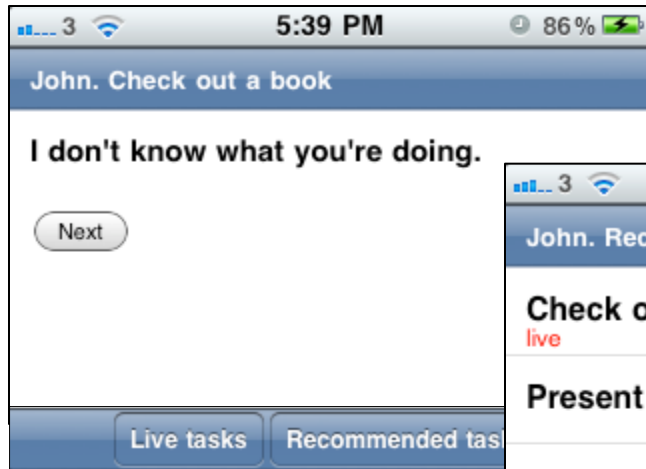
2. Proactive task guidance

- *Selecting 'Check out a book' will present the user step-by-step instructions to check out a book from the library*



Features of TASKOS (cont.)

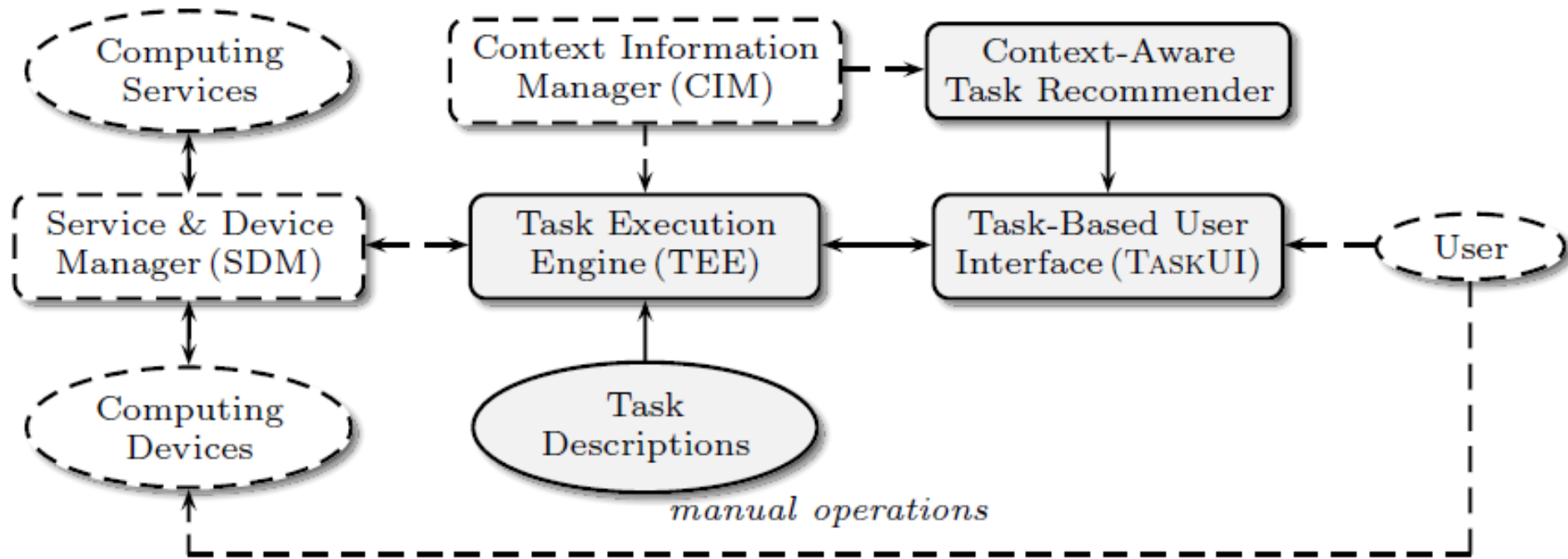
3. Task execution management: task status, multi-users, multi-tasks, service calls,...



```
<task id="SwipeIdCard">
  <dc:title>swipe your ID card over the card reader
</dc:title>
  <script>
    Packages.TaskOS.callService("scancardid");
  </script>
</task>

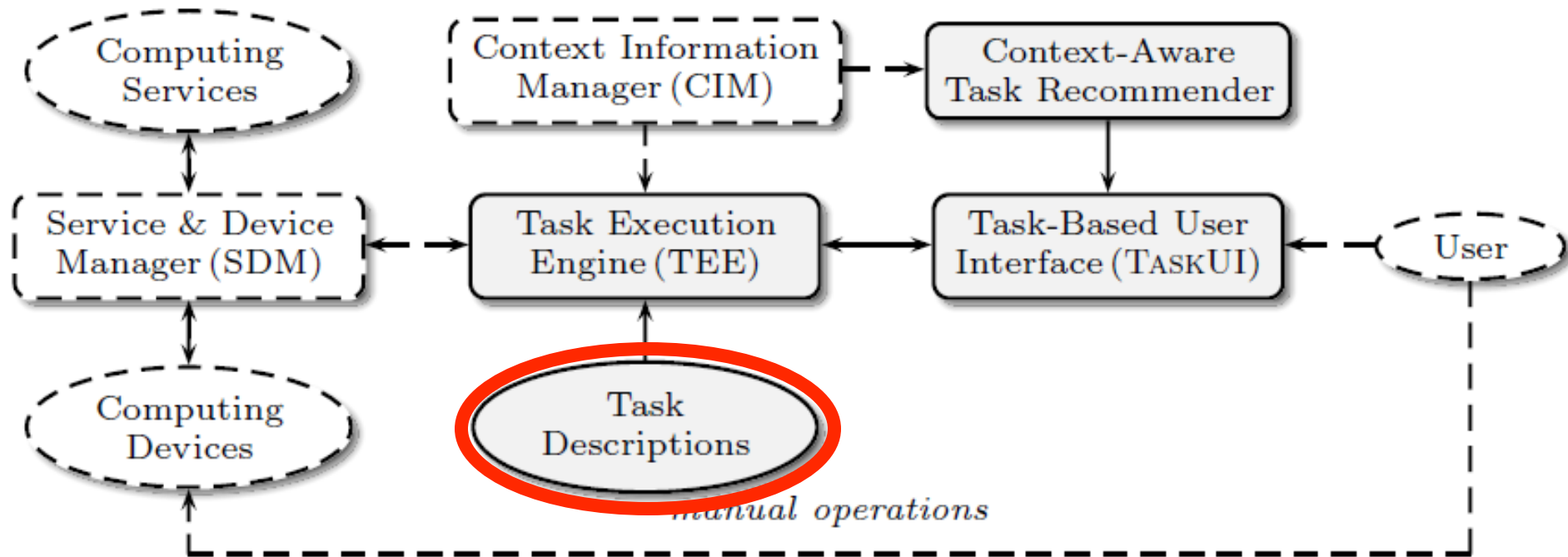
<task id="GetUser">
  <output name="user" type="String"/>
  <script>
    $this.user = Packages.Event.getValue();
    Packages.TaskOS.print("Your ID is " + $this.user);
  </script>
</task>
```

Conceptual architecture for TASKOS



The components with grey-fill and solid border are our focus in this paper.

Conceptual architecture (cont.)



- **Task descriptions:** XML specifications that describe how tasks are executed.
- **Task description language:** Syntax and semantics for describing tasks.

Example of a task description

```
<taskModel about="models:BookCheckOut"
  xmlns="http://ce.org/cea-2018"
  xmlns:dc="http://purl.org/dc/elements/1.1">

  <task id="Borrow">
    <dc:title>check out a book</dc:title>
    <subtasks id="borrowing">
      <step name="go" task="GoToCheckOutDesk"/>
      <step name="identifybook" task="IdentifyBook"/>
      <step name="identifyuser" task="IdentifyUser"/>
      <step name="checkout" task="CheckOut"/>
      <binding slot="$checkout.book" value="$identifybook.book"/>
      <binding slot="$checkout.user" value="$identifyuser.user"/>
    </subtasks>
  </task>

  <task id="GoToCheckOutDesk">
    <dc:title>go to the checkout desk</dc:title>
  </task>

  <task id="IdentifyBook">
    <dc:title>identify the book</dc:title>
    <output name="book" type="String"/>
    <subtasks id="IdentifyingBook">
      <step name="scanbook" task="SwipeBook"/>
      <step name="getbook" task="GetBook"/>
      <binding slot="$scanbook.external" value="true"/>
      <binding slot="$this.book" value="$getbook.book"/>
    </subtasks>
  </task>

  <task id="IdentifyUser">
    <dc:title>identify the borrower</dc:title>
    <output name="user" type="String"/>
    <subtasks id="IdentifyingUser">
      <step name="scanidcard" task="SwipeIdCard"/>
      <step name="getuser" task="GetUser"/>
      <binding slot="$scanidcard.external" value="true"/>
      <binding slot="$this.user" value="$getuser.user"/>
    </subtasks>
  </task>

  <task id="SwipeBook">
    <dc:title>swipe the book over the barcode reader</dc:title>
    <script>Packages.TaskOS.callService("scanbookid");</script>
  </task>

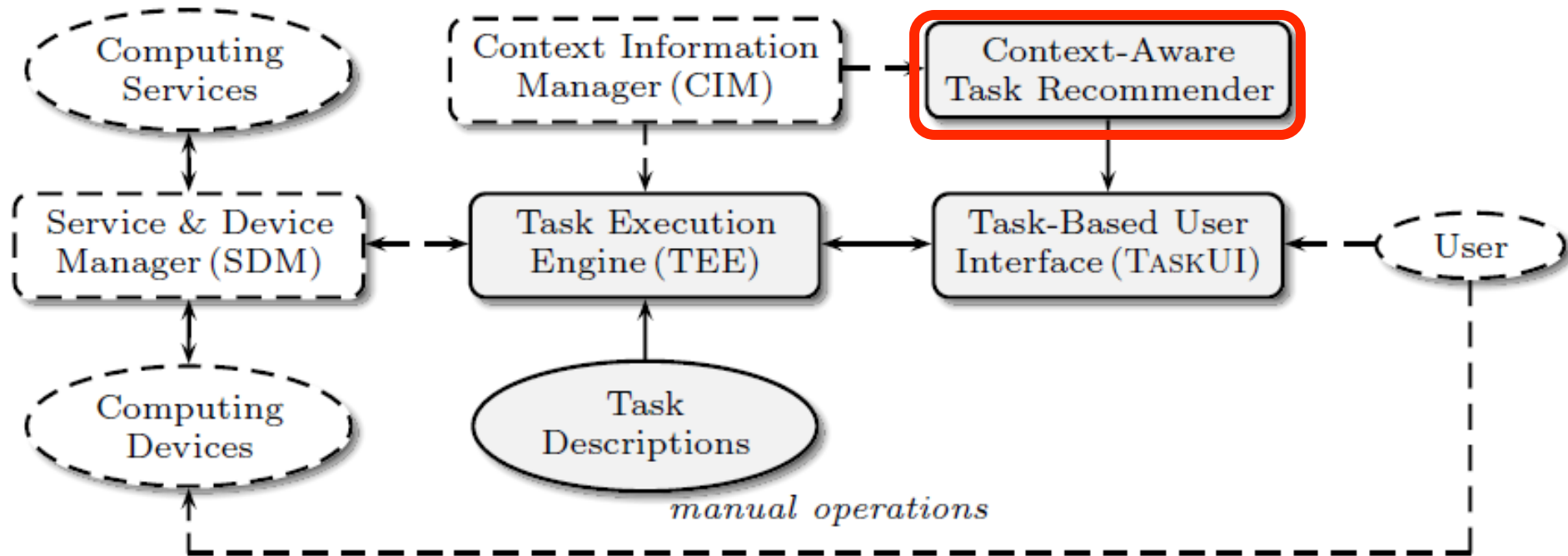
  <task id="GetBook">
    <output name="book" type="String"/>
    <script>
      $this.book = Packages.Event.getValue();
      Packages.TaskOS.print("The book ID is " + $this.book);
    </script>
  </task>

  <task id="SwipeIdCard">
    <dc:title>swipe your ID card over the card reader
  </dc:title>
    <script>
      Packages.TaskOS.callService("scancardid");
    </script>
  </task>

  <task id="GetUser">
    <output name="user" type="String"/>
    <script>
      $this.user = Packages.Event.getValue();
      Packages.TaskOS.print("Your ID is " + $this.user);
    </script>
  </task>

  <task id="CheckOut">
    <input name="book" type="String"/>
    <input name="user" type="String"/>
    <script>
      Packages.TaskOS.callService("checkoutbook " +
        $this.book + " " + $this.user);
      Packages.TaskOS.print("The book " + $this.book +
        " is checked out by " + $this.user + ".");
    </script>
  </task>
</taskModel>
```

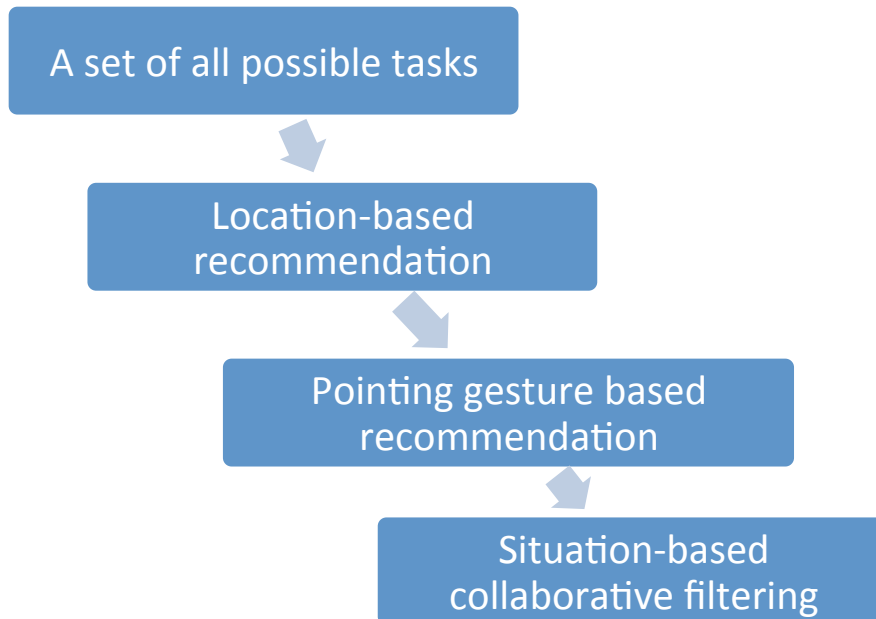
Conceptual architecture (cont.)



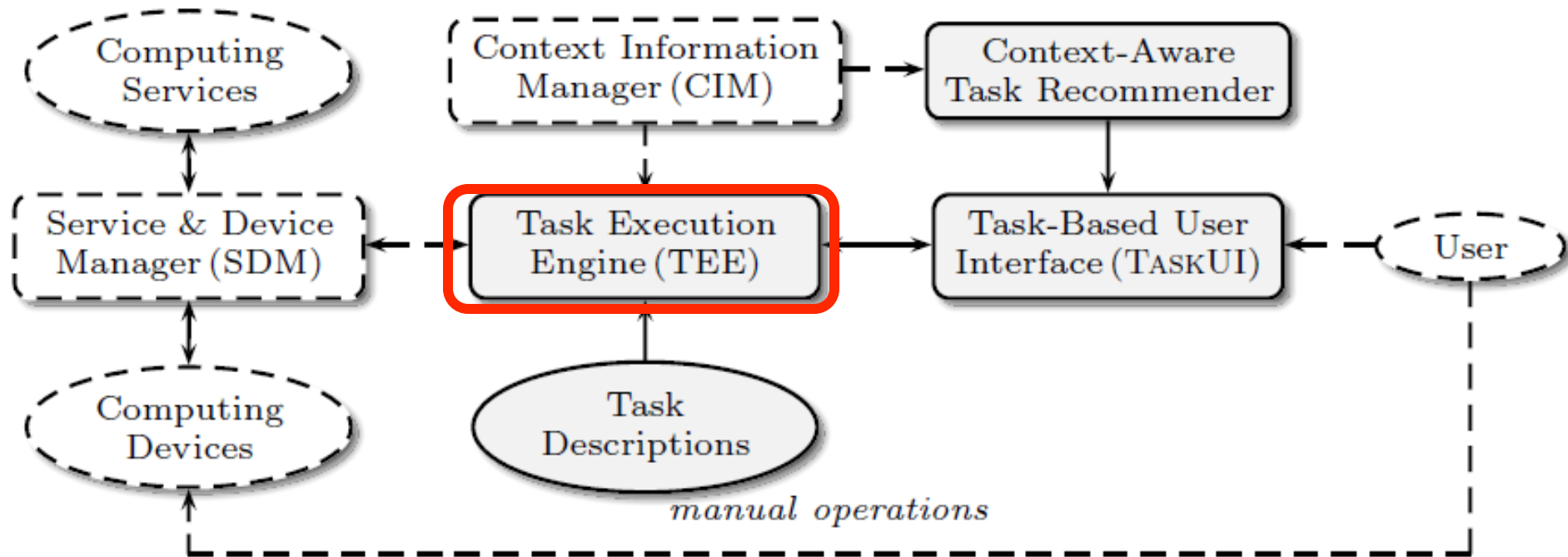
- **Task Recommender** aims to suggest users with possible and relevant tasks based on their context and available devices/services within a space.

Context-Aware Task Recommender

- A combination of the following methods:
 - Location based
 - Pointing gesture based
 - Collaborative filtering using situation similarity

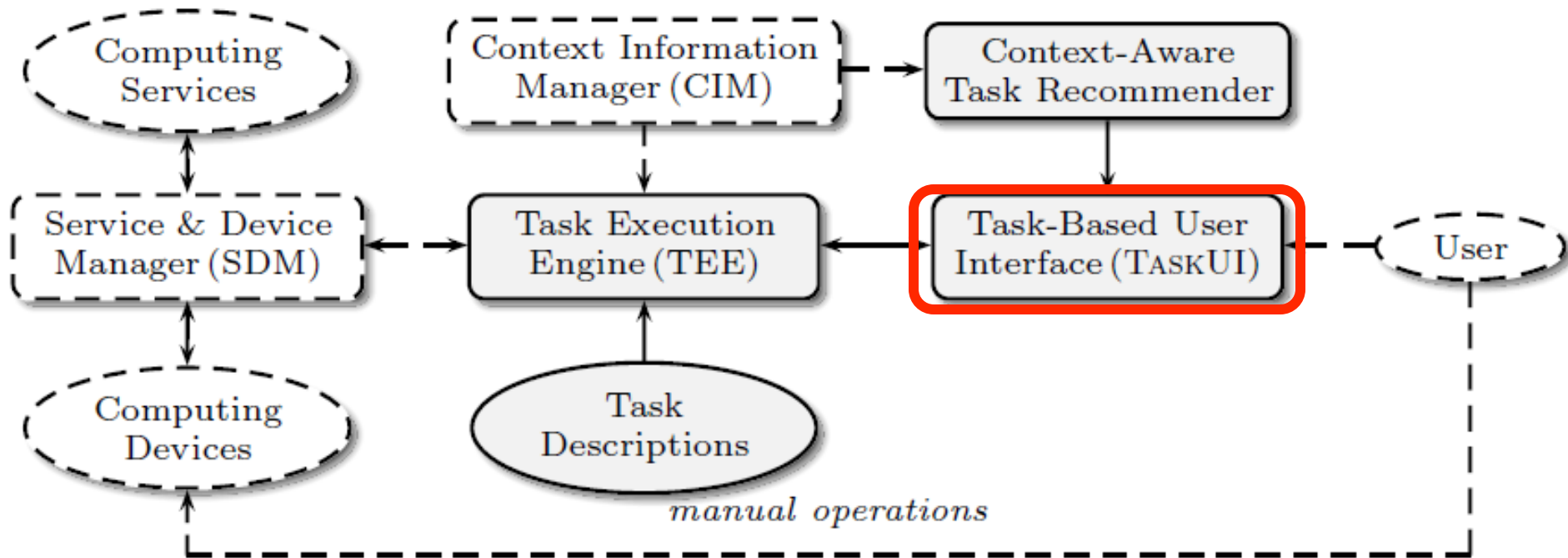


Conceptual architecture (cont.)



- **TEE** is to validate task descriptions, execute them, and manage their status.

Conceptual architecture (cont.)



- **TASKUI** is an interface for users to interact with TASKOS.

Advantages of TASKOS

Problem	Solution
Complexity of use	Task-based user interfaces & Proactive User Guidance
Invisibility & overload of features	Context-aware task recommendation
Inconsistency of UIs & task executions	Abstraction of task models

Related work

- **Situation-Aware Application Recommender** [Cheng *et al.* 2008]
 - Their system recommends single applications while our TASKOS recommends tasks which would involve many applications/devices/service.
- **InterPlay** [Messer *et al.* 2006]
 - Users express their tasks via a pseudo-English interface, then the system does the rest to achieve them.
 - The problem of InterPlay is that the users must learn how to express their tasks properly.
- **Huddle** [Nichols *et al.* 2006]
 - automatically generates interfaces for tasks which involve multiple devices, yet it only supports multimedia tasks that rely on data flows.
- **Task Computing Environment** [Masuoka *et al.* 2003]
 - represents a task as a composition of web services; hence TCE only supports autonomous, one-step, or batch tasks, e.g. “*exchange e-business card*”.
 - TCE does not support multi-step, interactive tasks, e.g. ‘*check out a book*’.
- **Aura** [Garlan *et al.* 2002]
 - supports the migration of tasks between environments. Our TASKOS does this too.
 - Since Aura’s task description language is designed only for capturing the status of tasks, it cannot describe envisioned tasks, which will be executed in the future.

Future work

- Evaluation:
 - A user study of TASKOS
- A graphical editor for authoring task descriptions
- Extend the task execution engine to support:
 - undo, redo, task-recording & replaying, and why menus.
- Algorithms and techniques for efficient and effective
 - storing, matching, retrieving, and recognizing task descriptions
- Address conflicts of resources in multi-user environments
- Mine task instructions from how-to websites to automatically construct task descriptions

References

- M. Weiser, "The computer for the 21st century," *Sci. American*, 3(265), pp. 94–104, 1991.
- C. Vo, T. Torabi, and S. Loke. Context-aware task recommendation. In *ICPCA-09*, Taiwan, 2009.
- Z. Wang & D. Garlan, "*Task-driven computing*," School of Computer Science, Carnegie Mellon University, Tech. Rep., 2000.
- D. Garlan *et al.* "Project Aura: toward distraction-free pervasive computing," *Pervasive Computing*, IEEE, 1(2), pp. 22–31, 2002.
- R. Masuoka *et al.* "Task computing – the semantic web meets pervasive computing," *The SemanticWeb*, pp. 866–881, 2003.
- D. Magnusson & B. Ekehammar, "Similar situations–similar behaviors? a study of the intraindividual congruence between situation perception and situation reactions," *J. of Research in Personality*, 12, pp. 41–48, 1978.
- A. K. Dey, "Understanding and using context," *Per. and Ubi. Computing*, 5(1), pp. 4–7, 2001.
- D. Cheng *et al.* "Mobile situation-aware task recommendation application," in *The 2nd Int. Conf. on Next Generation Mobile App., Services, and Tech.*, 2008.
- A. Messer *et al.* "InterPlay: A middleware for seamless device integration and task orchestration in a networked home," in *PERCOM'06*. 2006, pp. 296–307.
- J. Nichols, B. Rothrock, D.H. Chau, and B.A. Myers. Huddle: Automatically generating interfaces for systems of multiple connected appliances. In *Proceedings of the Symposium on User Interface Software and Technology*, pages 279–288, 2006.