1. Goal/Task Based User Interfaces

- **Basic concepts:** goals/tasks, recipes, plans
- **How used:**
  - hierarchical task analysis (modeling)
  - planning
  - plan recognition
Goal/Task Based User Interfaces

- System contains and *uses* (partial) model of what a person is trying to accomplish and how to operate the system to contribute
- Merely “contains” not enough, e.g., a searchable user manual
- But similar knowledge is contained in a *good* user manual --- “active manual”
- Example of “reflection” (often considered a hallmark of intelligence)

Goal/Task Based User Interfaces

- Kinds of knowledge system needs to represent and reason about:
  - *objects* (in the computer and/or real world) and their properties/relations
    - e.g., books, authors, ISBN numbers, etc.
    - e.g., window, scroll bar, text position, etc.
  - (primitive) *actions*/*events* and how they affect objects
    - e.g., lookup title in catalog, take book off shelf, etc.
    - e.g., open/close window, move cursor, etc.
Kinds of knowledge system needs to represent and reason about: [cont’d]

- typical user goals
  - e.g., borrow a library book
  - e.g., write a letter
- steps (decompositions) to achieve particular goals (under particular conditions)
  - e.g., find book; take book to checkout counter
  - e.g., start editor; write text; save file

Note “task” is used generically for primitive action or goal.

Somewhat more formally...

- Domain Model: \( \{ T, A, R \} \)
  - \( T \) object types
  - \( A \) (primitive) action types
  - \( R \) relations on \( T^* \)
- Task Model: \( \{ G, D \} \)
  - \( G \) goal types (built on \( T \) and \( R \))
  - \( D \) decomposition types (built on \( G \) and \( A \))
- A particular interaction instance:
  - \( T_1, T_2 \in T \); \( A_1,A_2 \in A \); \( G_1,G_2 \in G \); \( D_1 \in D \)
  - \( i,j \in T_1 \); \( k \in T_2 \)
  - \( G_1(i,k): D_1: < G_2(i,j): < A_1(i), A_1(j) >, A_2(k) > \)
Goal/Task Based User Interfaces

- Task modeling (analysis)
  - Defining the domain and task models for a particular application
  - Somewhat of an art—we’ll practice it later
    - finding right level of abstraction
    - what to ignore, what to make primitive
    - alternative groupings of lower-level actions/goals
  - “Achilles heel” of goal/task based user interfaces
    - special case of “knowledge acquisition bottleneck”
    - research towards automating using demonstration and/or learning techniques

Goal/Task Based User Interfaces

- Reasoning techniques
  - planning: finding a sequence of actions to achieve a desired state of the world
    - first principles
    - hierarchical task network
  - plan recognition: inferring plans or goals from observing actions
    - goal recognition
  - general inference: e.g., about domain relations
    - fast, incomplete (e.g., constraint propagation)
    - dependency-directed (e.g., TMS)
Goal/Task Based User Interfaces

- Planning tradeoffs
  - **First principles** ("classic")
    - requires complete domain model, i.e., the pre/postconditions of all actions accurately known
    - very flexible and general
    - many search algorithm variations: forward, backward, island, partial-order, hierarchical, etc.
    - task model not required, but can be used
  - **Hierarchical task network**
    - only the task model (predefined goals & decompositions)
    - task types can just be names (no pre/postconditions)
    - can use more domain model information if known

Research to synthesize the two approaches.

---

Goal/Task Based User Interfaces

- Plan recognition
  - reduces communication burden, e.g.,
    - observe $A_1(i) \rightarrow$ infer $G_2, G_1, D_1 \rightarrow$ suggest $A_2(j)$ next
    - versus
      - tell $G_1(i,k)$, tell $D_1$, observe $A_1(i) \rightarrow$ suggest $A_2(j)$ next
  - goal recognition: just $G_1, G_2$ (not $D_1$)
  - both kinds of recognition NP-complete in general
  - however, can often be done practically
    - incremental (in context of collaboration)
    - fall-back on communication
Goal/Task Based User Interfaces

- Interaction styles
  - pure advisor
    - user performs all actions (e.g., system does not have physical access to world)
    - system may not be able to directly observe actions (relies on user's reports)
    - system advises, critiques, etc. (fail soft)
  - purely automatic
    - system performs all actions
    - user provides toplevel goal (NB: difficulty of expression)
    - difficult to recover from undesired behavior
  - collaborative (mixed initiative)
    - best of both, but hardest to implement

Goal/Task Based User Interfaces

- Readings:
  - Lieberman & Espinosa, A Goal-Oriented Interface to Consumer Electronics using Planning and Commonsense Reasoning, IUI'06
  - ANSI/CEA-2018 Task Model Description (CE Task 1.0), 2007
    [Basis for semester project!]
Vote on CEA-2018

CEA-2018 Task Model Description
This usability problem is particularly acute when devices are connected in home networks, because achieving a single goal often requires understanding how to operate multiple devices.


Half of all “malfunctioning products” returned to stores by consumers are in full working order, but customers can’t figure out how to operate the device…
CEA-2018 Task Model Description

WG12 Participants:

Motorola      JVC      HBO
NAB          TCS    4HomeMedia
LonMarks     AT&T    DirectTV
Hitachi      LOYTEC  EchoStar
Alpine       Panasonic

1 SCOPE

A task model is a formal description of the activities involved in completing a task, including both activities carried out by humans and those performed by machines.

This standard defines the semantics and an XML notation for task models relevant to consumer electronics devices.
What we are not doing!

The standard does not define or restrict the actual appearance of a user interface or its detailed operation, and thus will not interfere with the preservation of brand identity via corporate logos or other means.

The standard does not depend on any specific home networking technology or infrastructure.

CEA-2018 Task Model Description

An Example Task Model

```
playMusic
  └── selectServer
      ├── selectNode
      │    └── selectRenderer
      │         └── selectPreset
      └── configureRenderer
          ├── connect
          └── play
```

selectRenderer
<taskModel
    xmlns=http://cet.org/ce2018
    xmlns:dc=http://purl.org/dc/elements/1.1
    xmlns:dcterms=http://purl.org/dc/terms>
    <dc:description xml:lang="en">CEA-2018 conformant sample task model description for
        playing music with UPnP AV devices and URC grounding</dc:description>
    <dc:creator>Gottfried Zimmermann</dc:creator>
    <dc:contributor>Charles Rich</dc:contributor>
    <dcterms:issued>2007-08-25</dcterms:issued>
    <dcterms:modified>2007-09-10</dcterms:modified>

    <task id="playMusic">
        <subtasks id="playMusicSteps" ordered="false">
            <step name="select" task="selectMusic"/>
            <step name="configure" task="configureRenderer"/>
            <step name="connect" task="connect" requires="select configure"/>
            <step name="play" task="play" requires="connect"/>
        </subtasks>
    </task>

    <task id="selectMusic">
        <subtasks id="selectMusicSteps">
            <step name="server" task="selectServer"/>
            <step name="browse" task="browse"/>
            <step name="node" task="selectNode"/>
            <binding slot="$browse.browseFilter" value='"*' />
            <binding slot="$browse.browseSortCriteria" value='"+dc:title"' />
            <binding slot="$play.connectionId" value="$connect.newConnectionId" />
        </subtasks>
    </task>

    <task id="configureRenderer">
        <subtasks id="configureRendererSteps">
            <step name="select" task="selectRenderer"/>
            <step name="preset" task="selectPreset" minOccurs="0" maxOccurs="1" />
        </subtasks>
    </task>

    <task id="connect">
        <input name="preferredConnectionProtocol" type="string" />
        <output name="newConnectionId" type="string" />
        <output name="error" type="ErrorDescription" />
    </task>

    <task id="play">
        <input name="connectionId" type="string" />
        <input name="playCurrentPlayMode" type="PlayMode" />
        <input name="playTransportPlaySpeed" type="PlaySpeed" />
    </task>

    <!-- external events -->
    <task id="transportStatusError">
        <output name="error" type="ErrorDescription" />
    </task>

    <task id="conNotifyContentFormatMismatch">
        <output name="error" type="ErrorDescription" />
    </task>

    <task id="conNotifyInsufficientNetworkResources">
        <output name="error" type="ErrorDescription" />
    </task>

    <task id="conNotifyUnreliableChannel">
        <output name="error" type="ErrorDescription" />
    </task>

    <task id="conNotifyUnknownConnectionError">
        <output name="error" type="ErrorDescription" />
    </task>
</taskModel>
Examples of Task-Based Applications

CEA-2018 will facilitate CE manufacturers developing a wide range of new capabilities to improve usability and customer satisfaction, such as:

- Network Command Menu
- Natural Language Access to EPG
- Intelligent Help Agent
- Task Personalization
- Intelligent Home Network Troubleshooting
- etc.

* Based on submissions to DLNA call for Far-Future Usage Scenarios

(1) Network Command Menu

- Copy
- Play
- Record
- Delete
- Stop

The Matrix in the living room
Star Wars in the bedroom
Survivor in the kitchen
CSI in the study
Numbers
Top Gear
(2) **Natural language access to Electronic Program Guide**

Record Survivor.

DVR

(3) **Intelligent Help Agent**

What do you want to do?

Copy a videotape to a DVD.

First, insert a blank DVD in the DVD recorder.

Ok, what next?

Push the button marked “Input 1” on the DVD recorder.

... etc.
(4) Personalized Tasks

Task: “Wake me up”
6:00 am – set thermostat to 70 F.
6:45 am – start coffee pot
7:00 am – ring alarm clock

“Wake me up tomorrow morning.”

Task Modeling Exercise