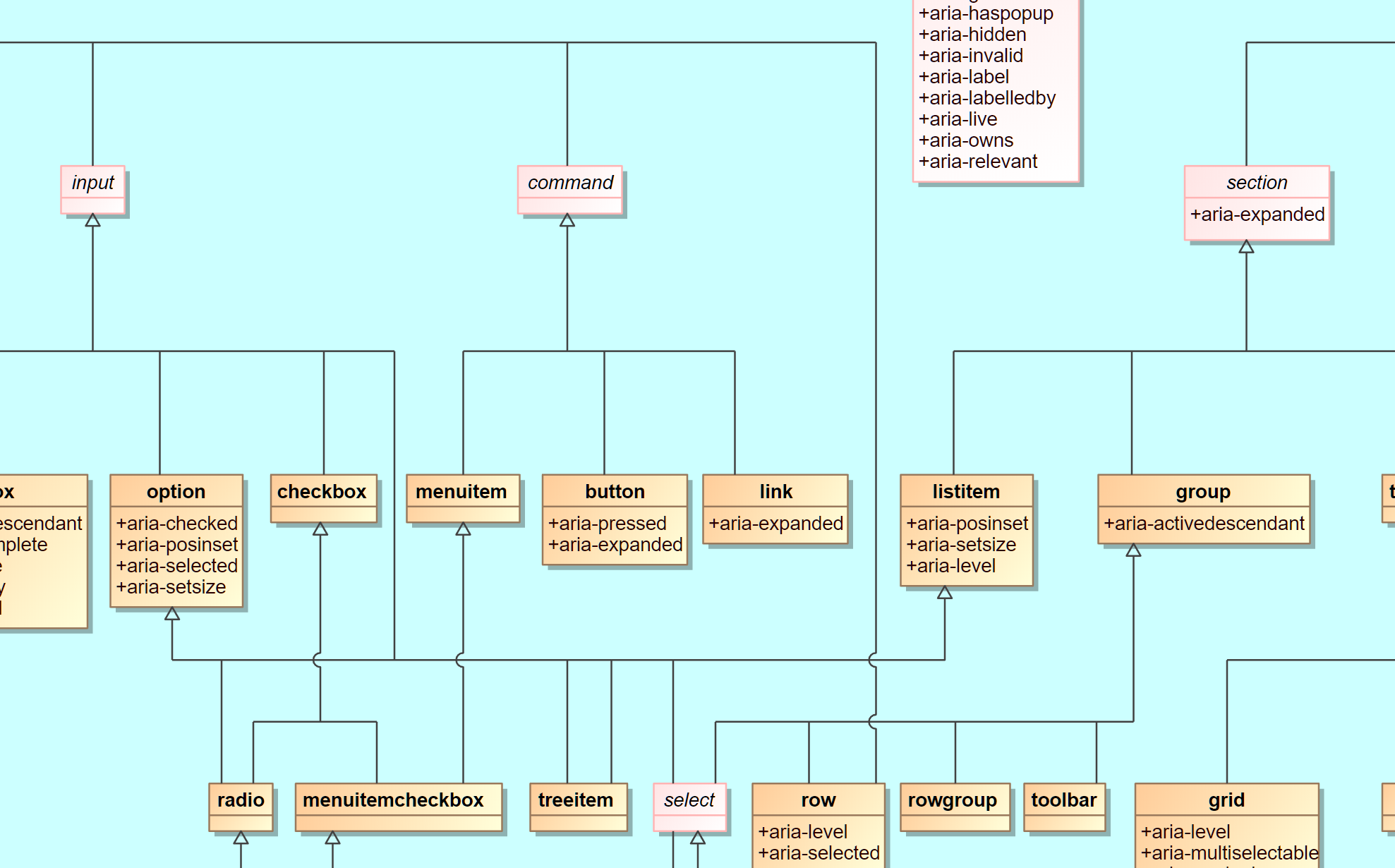
**ARIA needs more interactive container role(s)   
with focusable/active non-presentational sub content**

https://www.w3.org/WAI/PF/aria-1.1/rdf\_model.svg



Each class diagram is based on some fundamental **preconditions**. In the ARIA class diagram, the “left” input branch derived from generic **widget** (which combines input and command) brings activity in and the right branch generic **structure**. Also, the left branch consists concept-wise **of** **interactive content** whereas the right branch can have **active content** **in** the structure and is never “active” in a sense that it reacts on events and it is normally not focusable on its own.

In addition, it was a precondition that on mobile devices, active roles triggered by tap activation are always and strictly 1:1 also in the keyboard focus chain. The other way round, structural roles should NEVER react on TAP events since they normally get no keyboard focus.

While this separation made sense in the evolution of a first relatively simple class-based role model, actual platform and web development UI patterns considerably mix the different roles resulting in difficult “hybrid” roles that are not yet handled appropriately in ARIA.

This includes, but is not limited to, list-like containers that have actions on the container and contain focusable sub content. As a consequence, for content authors and assistive technology evaluating the role attribute from platform API there is a potential source of confusion what will be an “author error”, what is in a grey area and what will be the “right” behavior. For instance: When will a list item be active? By consequence, AT supporting the ARIA role model will not react predictably on list items with keyboard focus causing many strange and inconsistent issues.

**Problems**

**Listitems and Treeitems**

* listitem: not focusable (or not explicitely mentioned)
* treeitem: focusable (since it also derives from option)
* both class diagram concepts “clash” in a sense that in real life tree items MAY have interactive subcontent.

**Options**

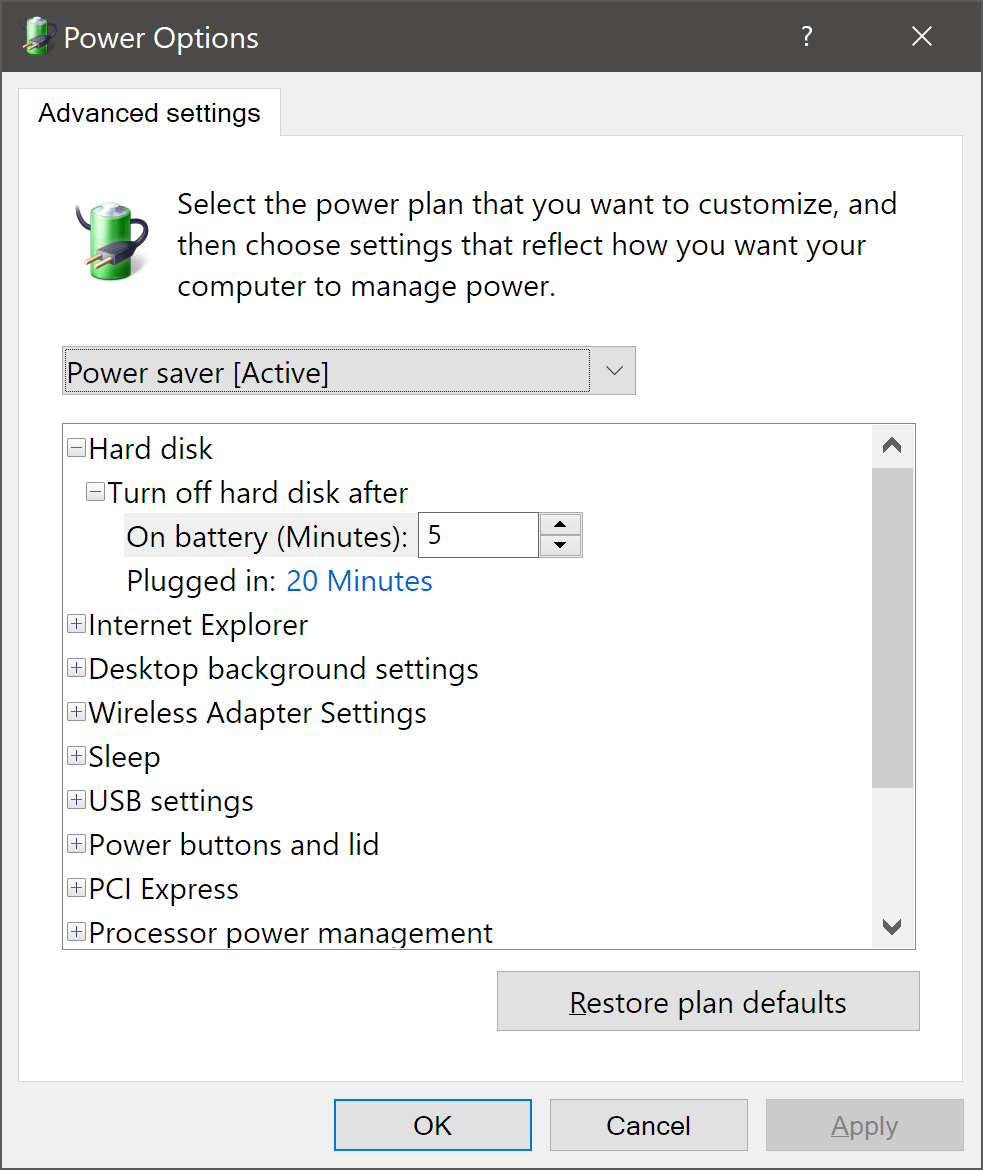
* no level info
* no possibility to group by property, just by structure
* Reason: option derives from generic input whereas listitem derives from section.

**Examples**

**Windows 10 Power Options Dialog**

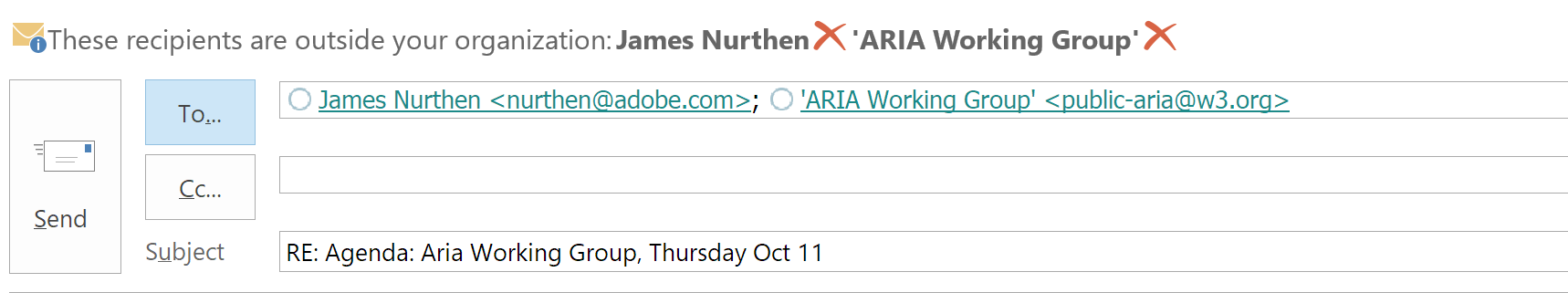
Hard Disk, Turn off hard disk after: On Battery (Minutes) Label + Numeric Spinner box

Within this are two active elements, the keyboard focus is by default on the label and arrow right acts as value selector for the item focusing an numeric spinner (spinbutton)



**Tokenizers**

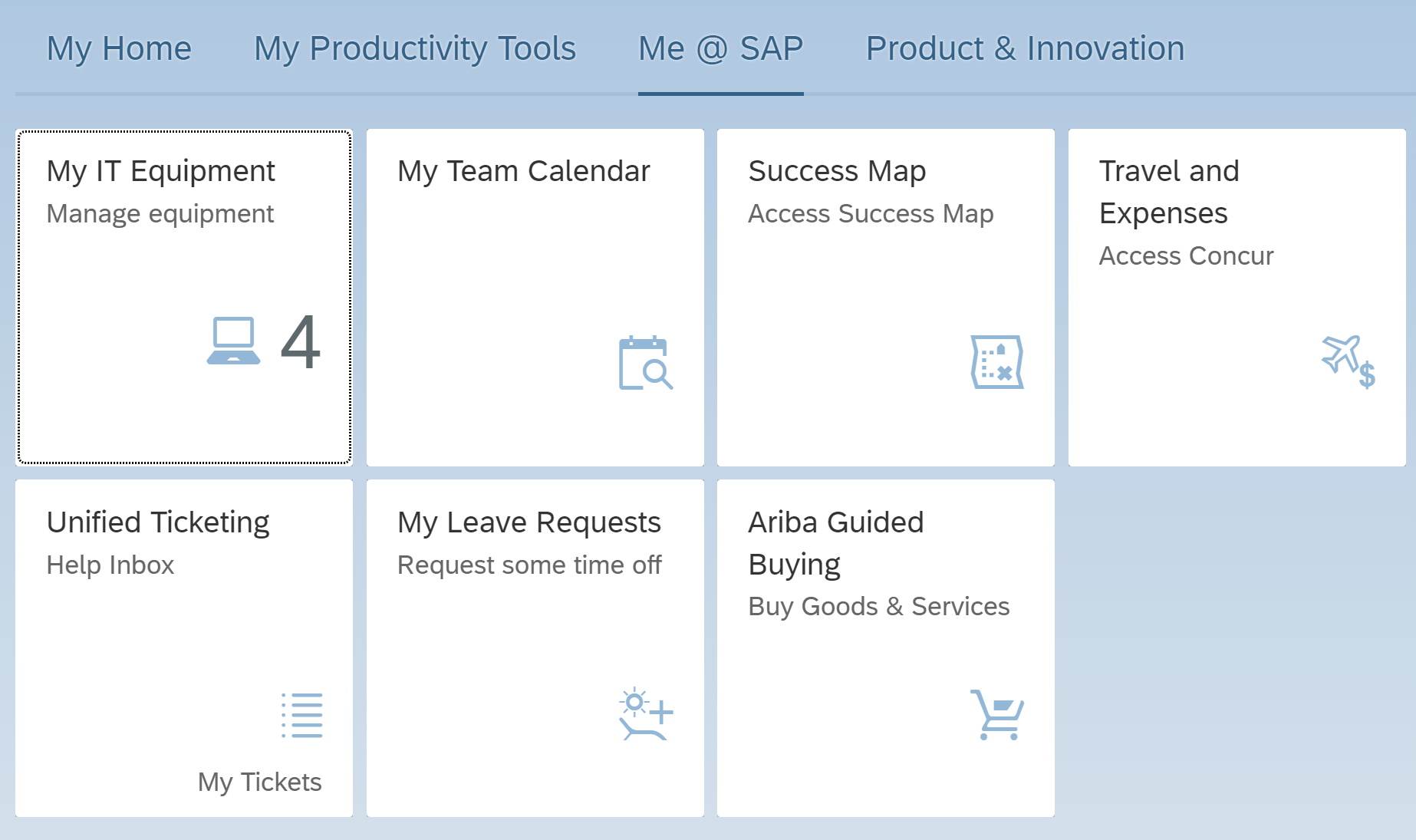
Windows Outlook “To:” address field with multiple names representing objects, not strings



Within an input there is not a simple value but a list of objects (of custom role, representing people, distribution lists etc.). Navigation between these takes place using left/right arrow keys. The individual items are NOT static but dynamic (react on activation) hence making the input to something EXTENDED and DERIVED from input (since still text input is possible and also deletion/adding of other list members).

**Interactive lists with active subcontent in listitems**

Lists consisting of elements with role=listitem with active subcontent are common in Web UI’s. These are also present on touch devices, where the large region of the list item is used for activation whereas the interactive content may hold links or other triggers to related information. No selection is possible here therefore role=option is not appropriate. Role listitem is problematic since listitem does not allow for keyboard focus. Roles like link or button are also inappropriate since these forbid non-presentational subcontent.



**Proposals**

* New role=composite/generic, role =“InteractiveList” with “InteractiveListItems”
* New “ListView” role (Example: Windows Explorer ListView)
  + Contains focusable ListItems (rows) containing focusable subitems
  + A ListView is NOT a [Treegrid](https://rawgit.com/w3c/aria-practices/treegrid/examples/treegrid/treegrid-1.html), these are different concepts
* Alternatives
  + Extend/subclass existing roles
  + Fallback strategies?
* NO Alternatives
  + Grid role for 1-column is NO alternative (!) for a List with active subitems
  + When you hear **grid** you expect x/y spatial navigation, NOT linear navigation