Abstract We describe three implementations of ‘middleware’ layers for incremental processing in dialogue systems, which take care of passing around and maintaining incremental information between the system modules. The implementations are all based on the model proposed by (Schlangen & Skantze 2009), but differ in certain important aspects.

The IU-Model (S&S’09) Basic notions:
- IU: Incremental Unit, minimal unit of information to be passed around between modules of processing system.
- Module: Consists of left buffer, processor, and right buffer.
- Operations:
  - add: new IUs are posted by one module for next one.
  - purge / revoke: IUs that were wrongly hypothesized are “taken back”. (E.g., “four” becomes “forty”.)
  - commit: module signals that it will not revoke IU.

InproTK Java-based implementation, but with provisions for allowing modules programmed in other languages (via OAA-bindings).
- event-driven left-to-right (bottom-up) processing, via even listeners
- communicates both edit messages (updates) as well as full buffers
- support for concurrent as well as sequential modules
- implemented in Java, integrated with Sphinx-ASR
- no information replication, access via grounded-in links, “intelligent” IUs
- comes with selection of modules, including ASR smoothing to avoid excessive hypothesis editing

Conclusions The approaches differ along certain dimensions.
- Strength of module coupling: IPAACA couples modules loosely, via brokering system; Jindigo enforces tight coupling via share IU-network. InproTK has provisions for both styles.
- Update passing, IU manipulation: IPAACA allows fully bi-directional IU manipulation, communicates updates.
  - jindigo packs updates as graph manipulations. InproTK communicates both delta and full buffer.
  - We are currently exploring ways to factor out common elements and encapsulate differences.

References & Acknowledgements
You can find the packages here: http://purl.org/net/Middlewares-SIGdial2010
Please download and try them out!