Modularity Framework

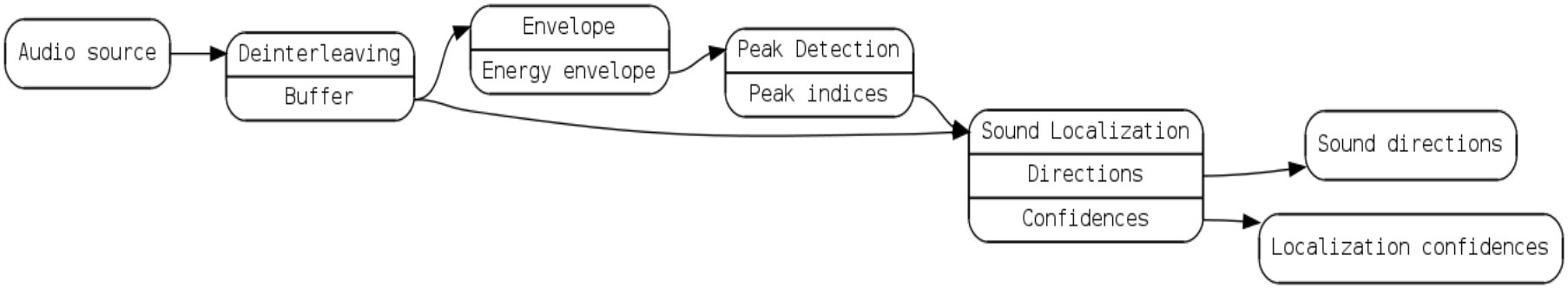
Grégory Rump

26/06/14

Modularity purpose

- Easy reworking/customization of parts of the algorithm
- Better understanding of the process flow
- Easier debugging by modular development
- Synchronized fetching of source data
- Factorization of common algorithm parts

Example (sound localization)



Modularity (main component)

- Input data synchronization

  The filter graph sink evaluation request is reversely propagated to the sources.

  Data sources are synchronized through simultaneous fetching

Modularity (main component)

- Create filters
  - Filter = algorithm applied to inputs to provide outputs
  - Composite filters
  - Input/output typing (one port can accept several types)
- Create processes (scheduled filter chains)
  - Created at runtime
  - Callback binding to sink

Modularity Scheduler

- Processes can be chained asynchronously
  I process B depends on the result of process A, B will use the latest result of A. This allows efficiency gain and ease of conception by loose synchronization between separate processes
- Process dependencies managed automatically

  If process B depends on the result of process A, but process A was not asked to run, process A will be scheduled anyway, transparently
- Scheduling priority between processes
  Allows to be sure some important embedded task is done even if resources are occupied by lots of processes
  Priority and frequency can be updated on the fly