Linkage graphs and what they look like

Stephen Kell

Stephen.Kell@cl.cam.ac.uk

Linkage graphs

Software has nontrivial static structure, and this is useful:

- re-use
- refactoring
- disaggregation
- **visualisation**

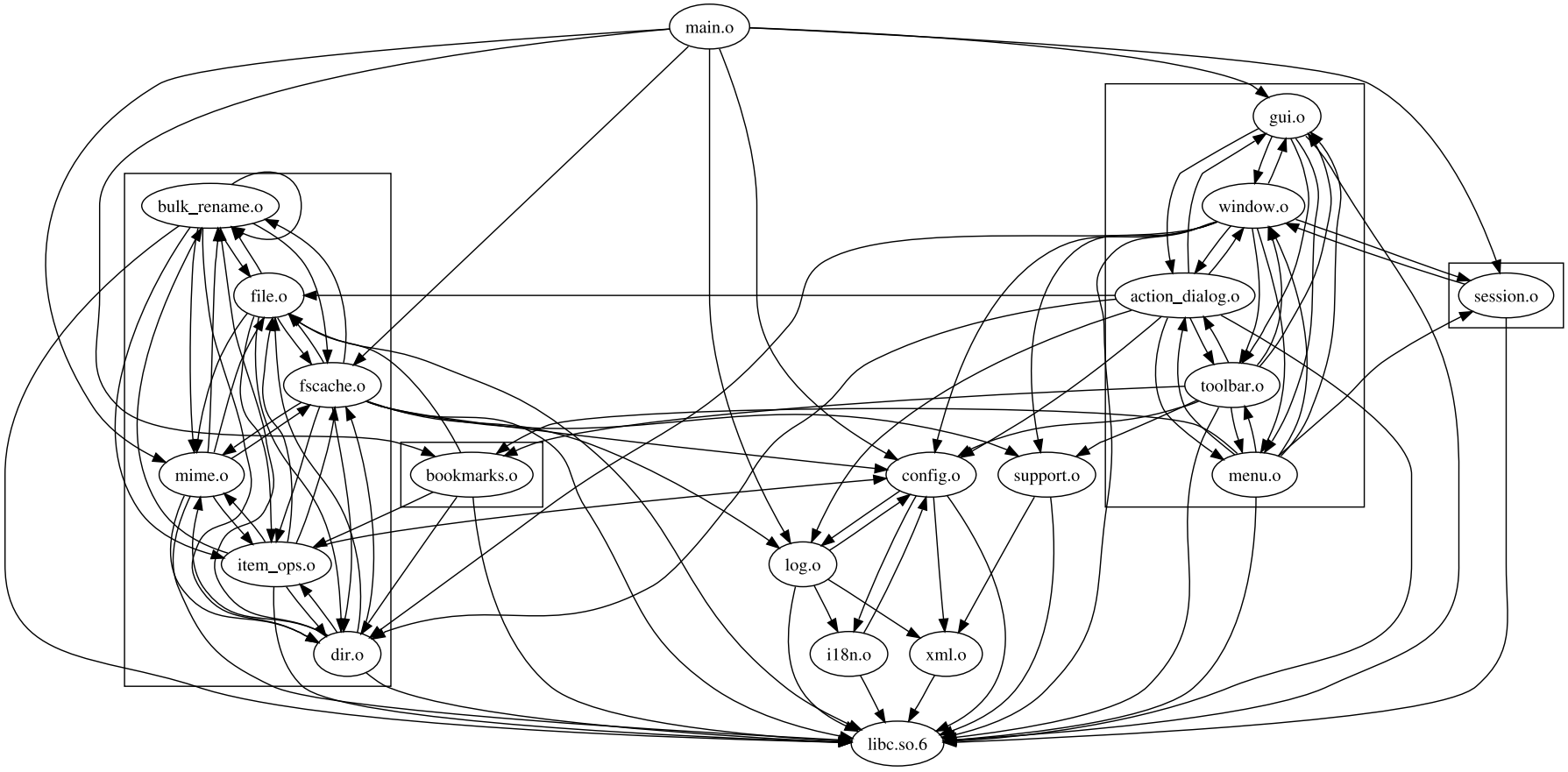
Problem: not all structure is made explicit by programmer.

- module import relation is coarse-grained

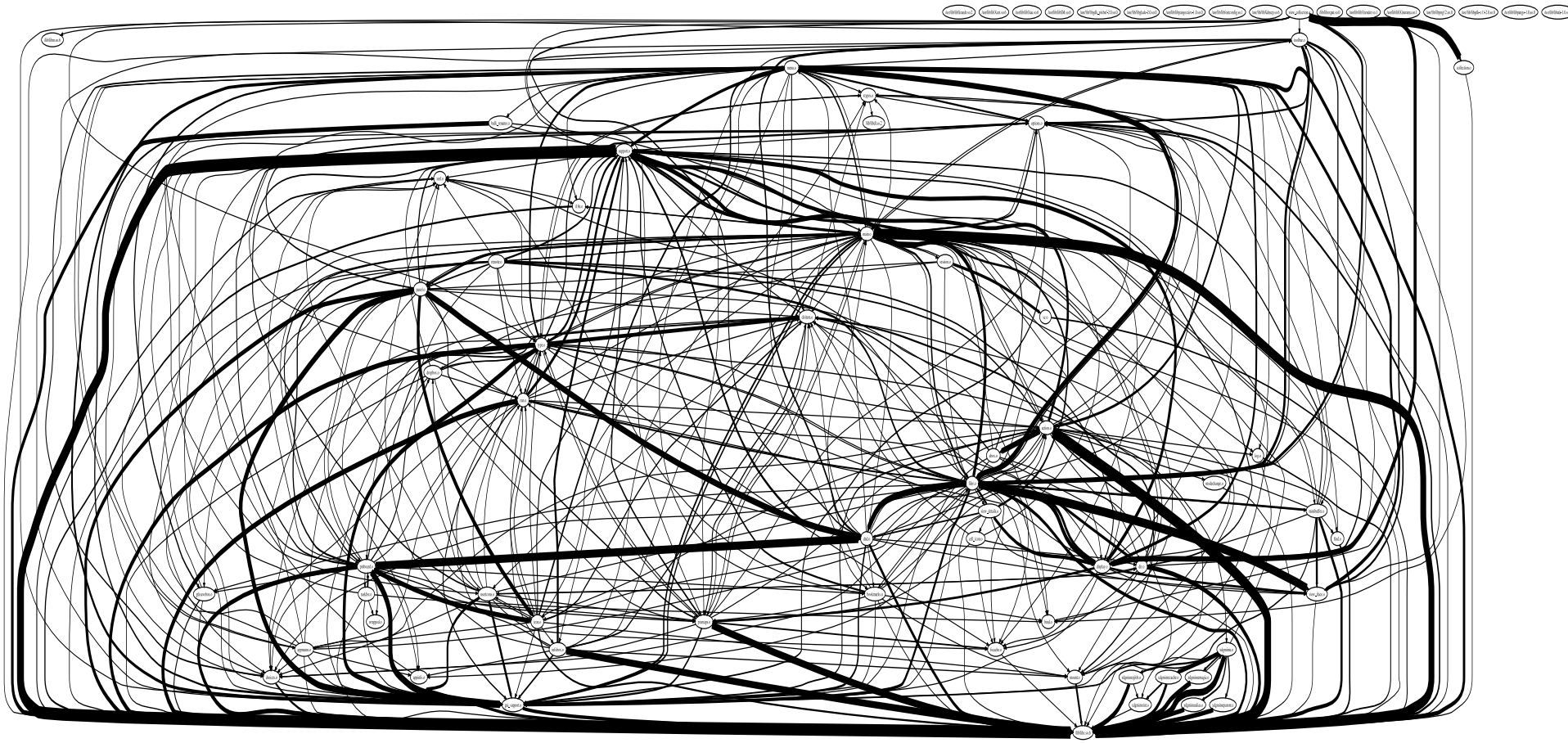
What does a linkage graph really look like? Let's find out:

- wrap `gcc` to generate dot file
- render with graphviz

You might expect. . .



A real example… (rox-filer)



Wanted: decomposed representation with fewer edges

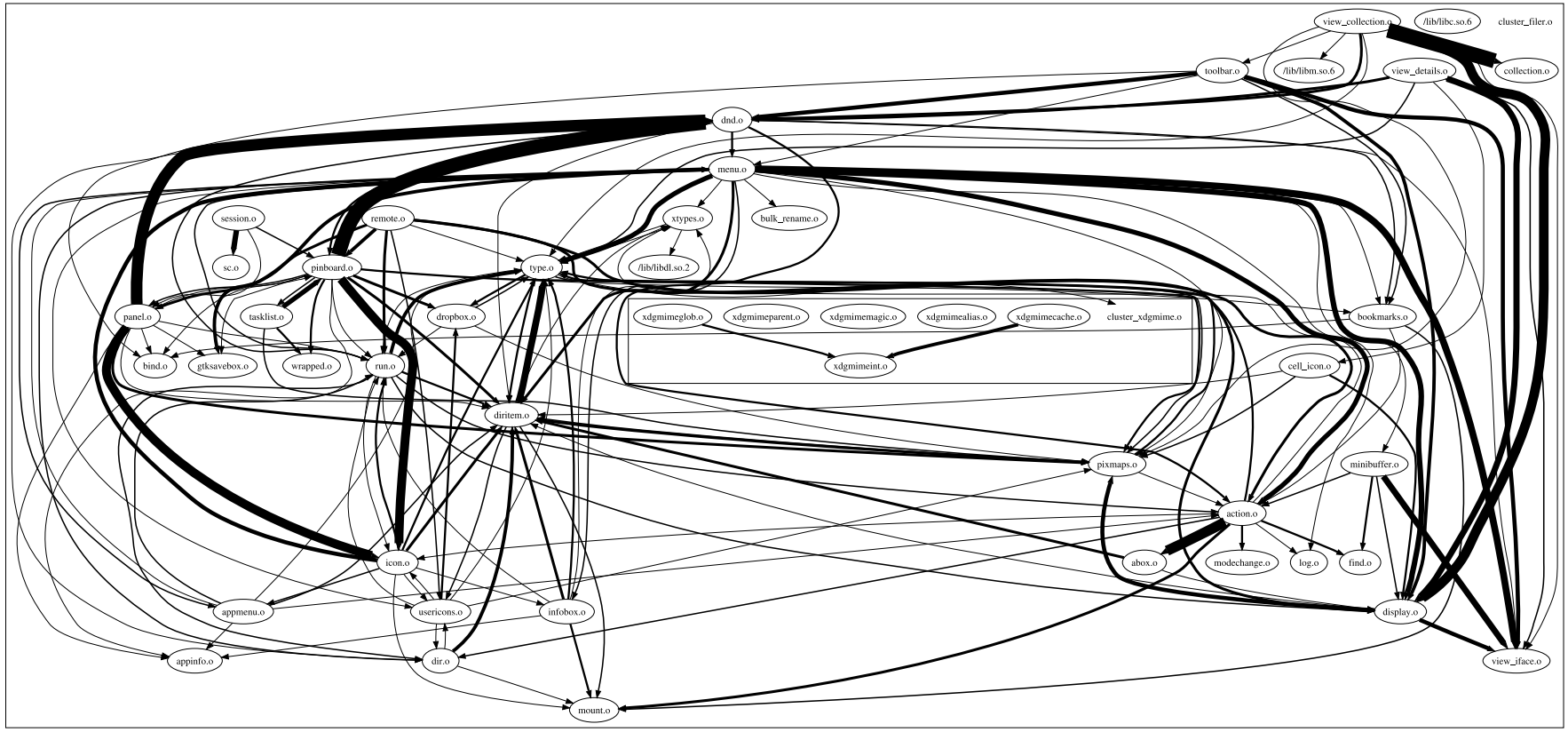
Graph decomposition? Sounds familiar

Some decomposition methods I'm aware of:

- strongly-connected components
  - can't apply recursively
  - strong connection is *too weak* a criterion
- community discovery
  - e.g. maximise Newman–Girvan modularity Q
  - doesn't help remove edges!
- my idea: edge aggregation
  - want draw one *aggregated* edge to/from a *cluster*...
  - ... instead of many single edges to/from nodes
  - might give poor Q, but good for visualisation

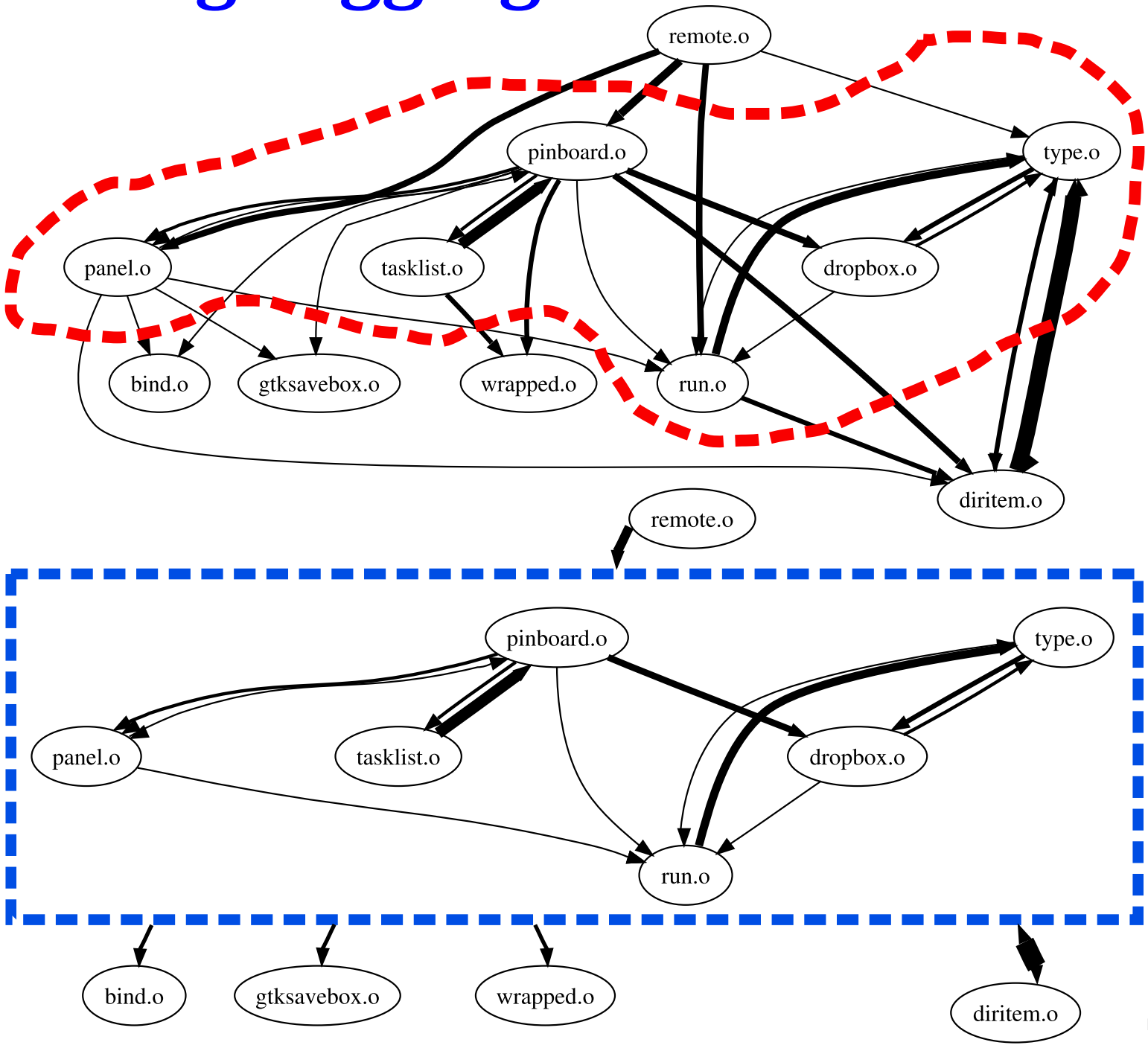
ROX filer after some ad-hoc clustering

After four rounds of head-scratching, it looks a bit better.



This was done mostly by deleting “pervasively-connected” nodes, together with their edges.

Edge aggregation in action



Formalising the process

Approach so far is ad-hoc. How do we make it systematic?

- define *goodness* of a cluster as *benefit* minus *cost*
- *benefit* is number of edges removed
- *cost* is trickier
- aggregating edges **entering** the cluster **from** node z:
  - cost is 0 if $z \rightarrow$ every node in cluster
  - else each non-z-connected node has a cost...
  - more hops away from $z \rightarrow$ greater cost?
  - not reachable from $z \rightarrow$ infinite cost? or just high?
- symmetrically for edges **leaving** the cluster **to** node z.

Cutting down the search space

Don't want exponential cost of considering all clusterings. Need a heuristic.

- very crude first cut: *gateway sets*
- intuition: connectivity distribution is asymmetric
  - often have unique entry node ("interface module")
  - rarely have unique exit node
- $GatewaySet(z)$ is set of nodes reachable only through z
- gateway nodes have finite (usually small) *entry* cost
- prune dfs descendent tree to find reasonable *exit* cost
- problem: may not have unique entry node...

That's all for now. Ideas welcome!

Spare slide: tail-end example