Cradle to Grave: Complete Evolution of the LS/2000 Software Architecture*

James R. Cordy, [1] Thomas R. Dean, [1]
Andrew J. Malton, [2] Kevin A. Schneider [3]

Legasys Corporation
Kingston, Canada

[1] Queen's University, Canada
[2] eSentire Inc., Canada
[3] University of Saskatchewan, Canada

* Original version Sept 1997

Why ?

- A chance to capture the true evolution of a large software system, from inception to production to maintenance
- Documentation and archive of architectural decisions and reasons
- Experiment with using Dean/Cordy software architecture syntax
- Learn from experience

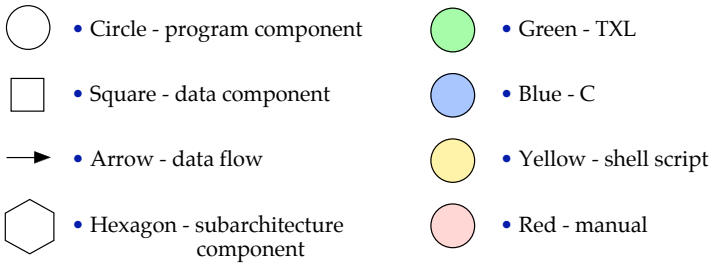
What (was) LS/2000 ?

- World's leading automated Year 2000 conversion system
- Licensed to: IBM Global Services, EIS, PRT, Format 2000 Ltd., LS 2000 Pty Ltd., ...
- Used by: Every major bank in Canada, major retailers, insurance companies, governments, airlines, health industry
  - in Canada, U.S., U.K. and abroad
- Key tools and technologies (Dean et al., ICSM 2001):
  - robust parsing (Barnard & Holt 1982),
  - source transformation (TXL, Cordy et al. 1988),
  - design formalization using factbases (Schneider & Lamb 1992)
  - design recovery using source markup (Cordy & Schneider 1995)

An Evolutionary Tour

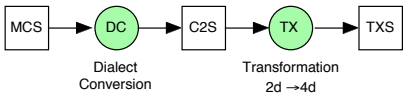
- What we are going to do is walk through the evolution of the LS/2000 process architecture, step by step
- Consider the changes in requirements and customer expectations and their effect on the architecture
- Trace changes in technology and tools
- Hopefully learn something
- Remember:
  - Continuously in production and delivering results as evolves
  - Continuously scaling up, from 100 Kloc to 16 Mloc programs, 10 Kloc to 1 Mloc single source files

Basic DCA Notation



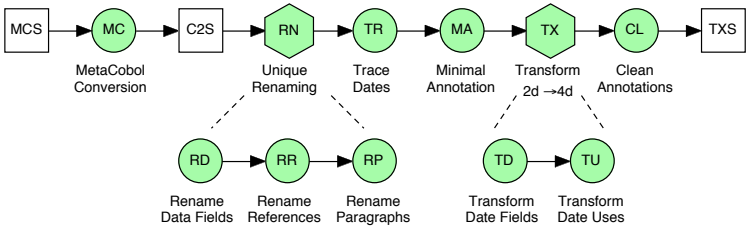
- Circle - program component
- Square - data component
- Arrow - data flow
- Hexagon - subarchitecture component
- Green - TXL
- Blue - C
- Yellow - shell script
- Red - manual

January 1995 : Original Design



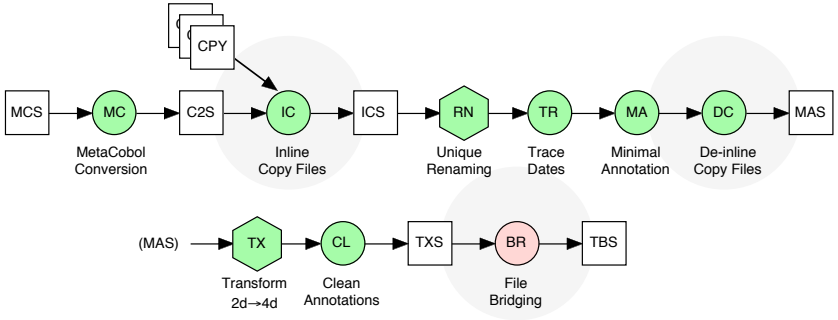
- one step dialect conversion, any Cobol (C1, C2, MetaCobol) → C2
- one step Year 2000 transformation, 2 digit to 4 digit fields, direct pattern transformation
- one file at a time, no inter-file analysis
- same process for COPY members (i.e., header files), no inter-file analysis

April 1995 : First Viable Implementation



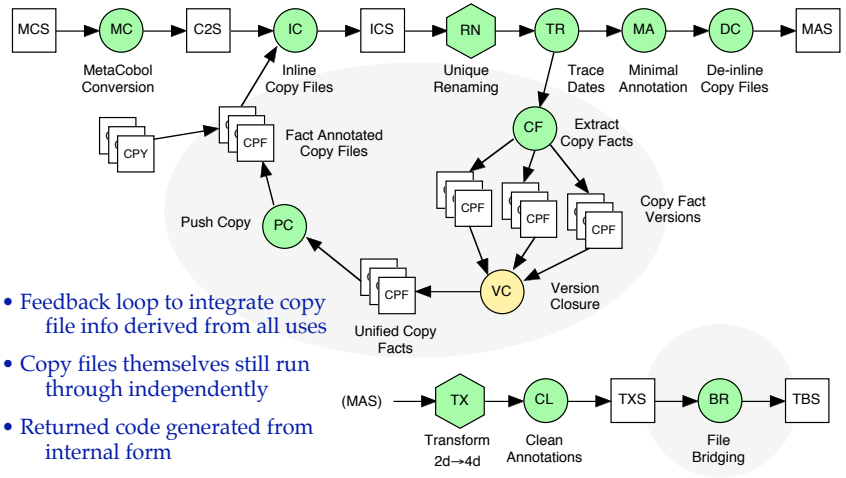
- one step dialect conversion, MetaCobol → C1/C2
- unique renaming to disambiguate unqualified and clashing names
- four step Year 2000 transformation :
  identify dates by naming convention,
  trace dates by operation,
  annotate dates in source,
  transform date fields and operations by pattern
- one file at a time, no inter-file analysis
- COPY files same process, no way to reconcile with programs

June 1995 : Inline Copy Files for Trace



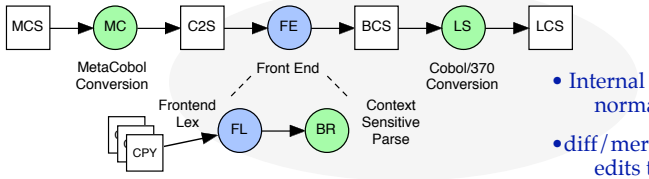
- Copy (include) files inlined to assist in date tracing, but still processed separately, no inter-file analysis
- File bridging (2d → 4d, 4d → 2d input/output) done by hand

September 1995 : Inter-file Copy Analysis

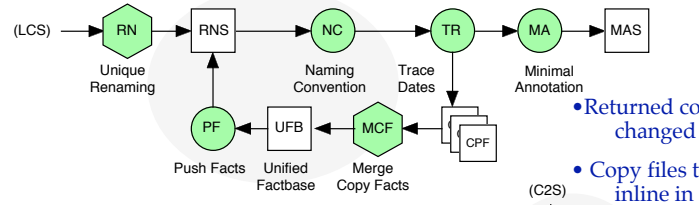


- Feedback loop to integrate copy file info derived from all uses
- Copy files themselves still run through independently
- Returned code generated from internal form
- File bridging automated

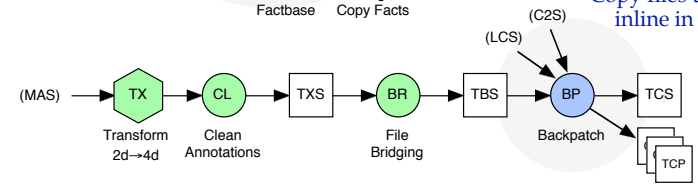
December 1995 : Preserve Client Format



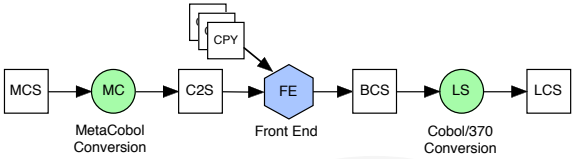
- Internal LS/Cobol format - normalized, easy to process
- diff/merge derives minimal edits to original C2S source



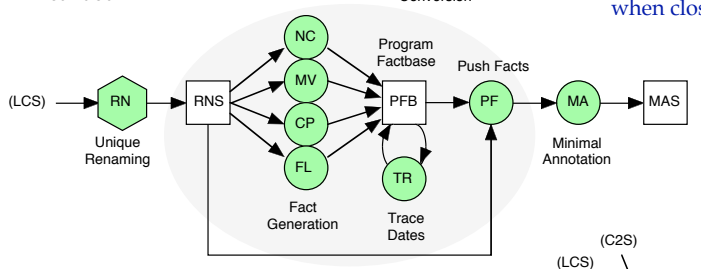
- Returned code minimally changed original
- Copy files transformed inline in context

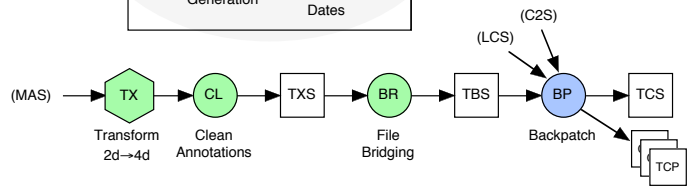


February 1996 : Separate Factbase

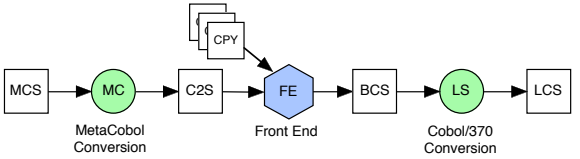


- Facts separated from source as a single database
- Date tracing done at fact level
- Results reflected into source when closure reached

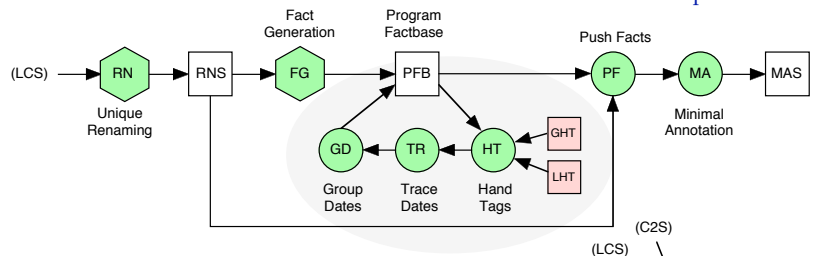


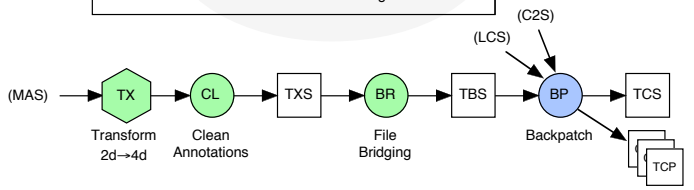


March/April 1996 : Add Analyst to Trace Cycle



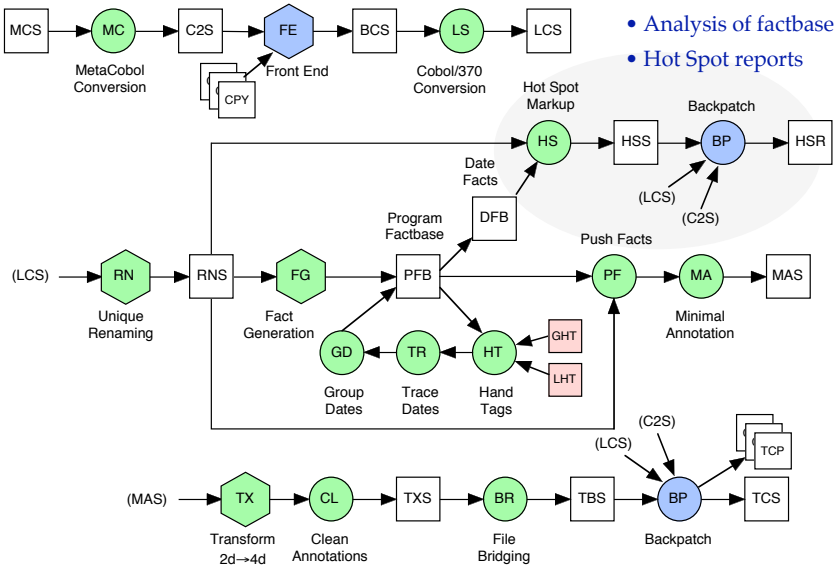
- Trace cycle allows analyst interaction
- Table-driven client and application-level customization
- Group date inference





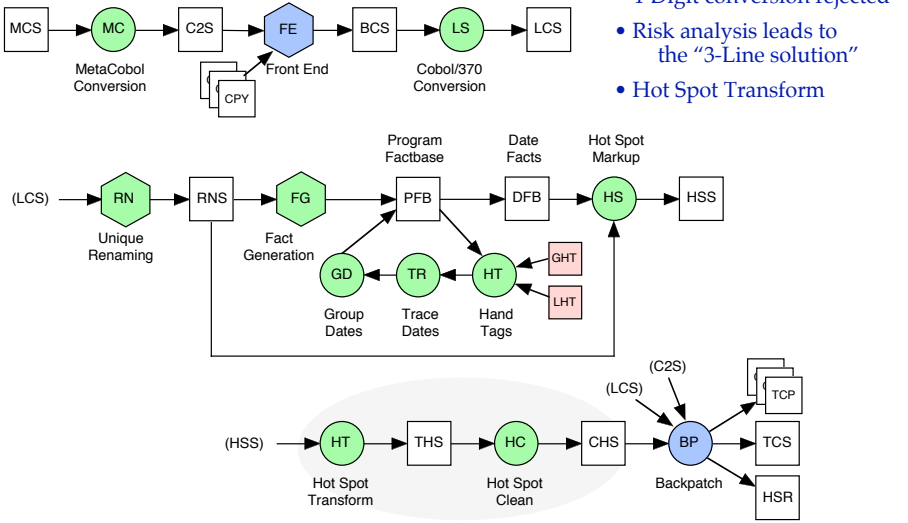
June 1996 : Hot Spot Reporting

- Year 2000 "Hot Spot" identification
- Analysis of factbase
- Hot Spot reports



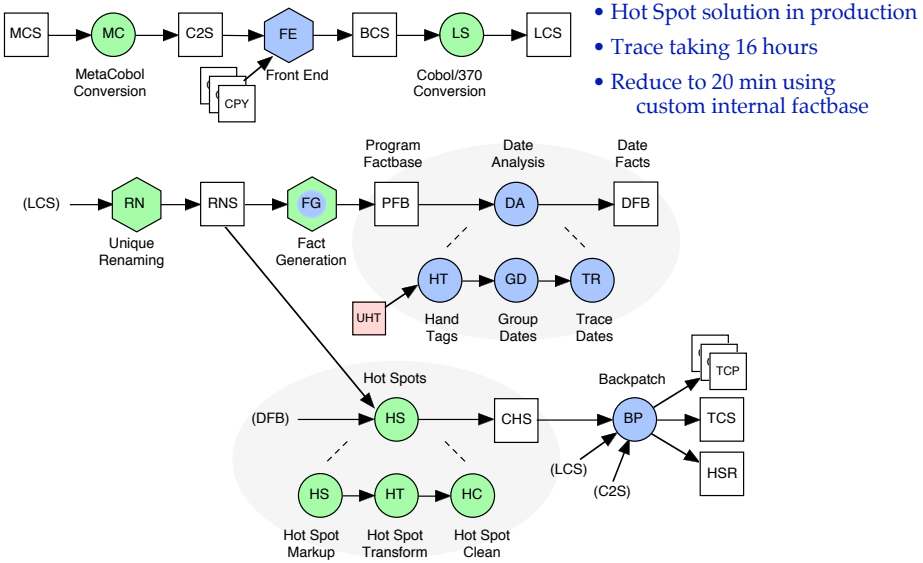
August 1996 : The "Disaster Meeting"

- 4-Digit conversion rejected
- Risk analysis leads to the "3-Line solution"
- Hot Spot Transform



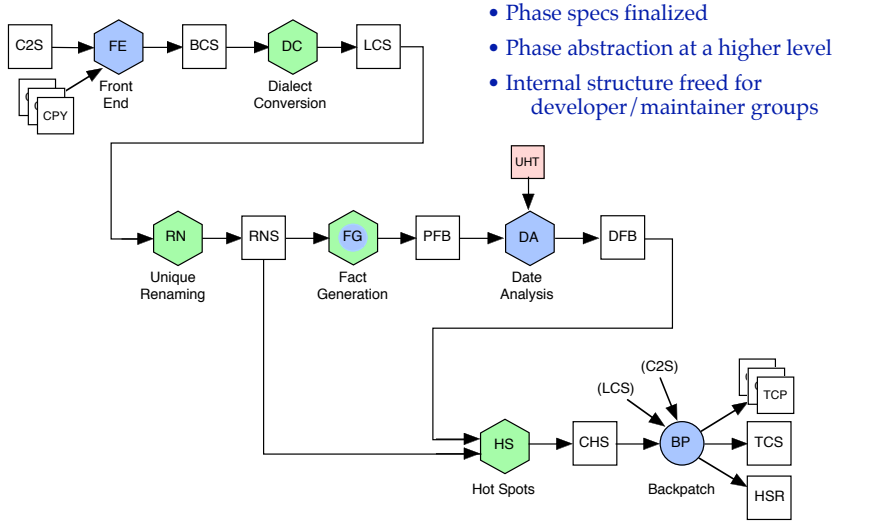
Oct 1996 - Jan 1997 : Production Tuning

- Hot Spot solution in production
- Trace taking 16 hours
- Reduce to 20 min using custom internal factbase



March 1997 : Licensed Export

- Phase specs finalized
- Phase abstraction at a higher level
- Internal structure freed for developer/maintainer groups



LS/2000 in Production

- Process cloned and adapted to PL/I, RPG, Natural, Fortran, CICS, IMS, SQL, Adabas, ...
- Generalized to other business types and concepts - LS/AMT (flight booking systems, health systems)
- Processed over 4.5 Gloc of code 1995-2001

So what have we learned ?

- Architectural changes driven not by design or specification changes, but by rapid changes in user-end requirements and expectations
  - use clear, small, independent stages for easy replacement
- Higher level partitions and abstractions cannot be envisioned at design time, rather evolve into existence as software system matures
  - avoid integration of tools and techniques for easy reconfigurability
- Academic analysis tools and techniques (TXL, fact-based DR) can be practical and scale up (but a lot of work!)
  - replace with custom solutions as stages mature

Questions?