Distr.
GENERAL

CES/SEM.47/18 (Summary)
17 January 2002

Original: ENGLISH

STATISTICAL COMMISSION and ECONOMIC COMMISSION FOR EUROPE

COMMISSION OF THE EUROPEAN COMMUNITIES

CONFERENCE OF EUROPEAN STATISTICIANS

EUROSTAT

Joint UNECE/Eurostat Seminar on Integrated Statistical Information Systems and Related Matters (ISIS 2002)
(17-19 April 2002, Geneva, Switzerland)

Topic III: Object-oriented technologies, component architecture

ARCHITECTURAL DIRECTIONS AT STATISTICS NETHERLANDS

Invited paper

Submitted by Statistics Netherlands [1]

Summary

1. Recent changes into a process-oriented organization have directed software development at Statistics Netherlands towards building fewer but bigger applications. Primary development capacity and responsibility resides with the statistical divisions. The central technology division supplies capacity for special projects or to compensate for shortages.

2. The software environment is Microsoft with Visual Studio and COM+ but without Transactionserver. Private import or procurement of software is not allowed because of the continuity problems this causes. Use of databases (SQLserver) is encouraged and so is the use of the OLAP engine.

3. Software architecture is not only influenced by the development environment but also by the existing standard software created at Statistics Netherlands. This software is increasingly becoming available as component packs. Together with this "componentization" of the standard software a new kind of middleware, Cristal, is being developed. Cristal is a class tree component that offers standardized I/O in a statistical oriented memory structure with standard load/unload components.

4. Principal guidelines in software architecture are the use of multi tier architecture and use of components whether they are standard components or not. The reuse of code, especially in component form, is promoted.

[1] Prepared by Marton Vucsan (mvcn@cbs.nl).

5. The administration of components, especially a centralized administration, has proven to be difficult. Maintenance and continuity is difficult to guarantee without assigning a regular maintenance project etc. This discourages the central deployment of small components. Because of the runtime binding of COM+. DLLhell can only be avoided by assigning new identifiers to modified or corrected components.

6. The use of components is becoming more commonplace with components that are more focused on performing complex statistic functions. Earlier experiences with code sharing were a disappointment caused by the fact that components or reused code pieces were mostly small and very technical. Developers tended to create these little pieces of software themselves.

7. For the design of components three sources can be identified. First on a local scale two or more applications can share code, which is then made into a component. Second, standard software has parts that deserve to be available on their own. Third, a new methodological process or theory is made into a component.