EUROPEAN PATENT OFFICE
U.S. PATENT AND TRADEMARK OFFICE

CPC NOTICE OF CHANGES 1912

DATE: AUGUST 1, 2026

PROJECT DP12940

The following classification changes will be effected by this Notice of Changes:

Action	**Subclass**	**Group(s)**
DEFINITIONS:		
Definitions Modified:	G06F	11/14, 11/1446, 11/1471, 11/1474, 11/1479, 11/1482, 11/1487, 11/1489, 11/1492, 11/1497

No other subclasses/groups are impacted by this Notice of Changes.

This Notice of Changes includes the following *[Check the ones included]:*

1. CLASSIFICATION SCHEME CHANGES
   - ☐ A. New, Modified or Deleted Group(s)
   - ☐ B. New, Modified or Deleted Warning(s)
   - ☐ C. New, Modified or Deleted Note(s)
   - ☐ D. New, Modified or Deleted Guidance Heading(s)
2. DEFINITIONS
   - ☒ A. New or Modified Definitions (Full definition template)
   - ☐ B. Modified or Deleted Definitions (Definitions Quick Fix)
3. ☐ REVISION CONCORDANCE LIST (RCL)
4. ☐ CHANGES TO THE CPC-TO-IPC CONCORDANCE LIST (CICL)
5. ☐ CHANGES TO THE CROSS-REFERENCE LIST (CRL)

2. A. DEFINITIONS (Modified)

G06F11/14

Replace: The existing Definition statement with the following updated statement.

Definition statement

This place covers:

Prophylactic additional saving-related measures like check-pointing, backing-up or state copying, which are performed before the occurrence of a fault in order to be able to recover or restore (at least partially) in case a fault occurs in the future, and which do not rely on hardware redundancy.

Corresponding error correction means comprising reverse operations of restoring or recovering or rolling back, which are performed after the occurrence of a fault, e.g. redoing.

Error detection mechanisms, after the occurrence of a fault, which include retrying an operation in conjunction with a counting or time-out scheme. These mechanisms are often used to overcome a transient error.

G06F11/1446

Replace: The existing Definition statement with the following updated statement.

Definition statement

This place covers:

Backup performed either at file or data block level. This includes backing up of any type of file, independent of the data it contains.

Means that ensure the data to be saved (as backup copy) is consistent (i.e. represents a meaningful state), especially if the copy is made of data within a distributed system.

For the purposes of classification:

- backups differ from archives in the sense that archives are the primary copy of data whereas backups are a secondary copy of data.
- data backup systems differ from fault-tolerant systems in the sense that data backup systems assume that a fault will cause a data loss event, whereas fault-tolerant systems do not.

G06F11/1471

Replace: The existing Glossary of terms table with the following updated table.

Glossary of terms

In this place, the following terms or expressions are used with the meaning indicated:

logging	recording physical or logical changes to stored persistent data to allow a system to recover from crashes or other errors and maintain the stored persistent data in a consistent state
journal	a chronological record of data processing operations. It is considered equivalent to logical logging.

G06F11/1474

Replace: The existing Definition statement with the following updated statement.

Definition statement

This place covers:

General recovery techniques of transactional systems, recovering from errors within transactions that create, update or modify data.

G06F11/1479

Replace: The existing Definition statement with the following updated statement.

Definition statement

This place covers:

All error detection or fault masking techniques implemented by software means, i.e. where the fault tolerance does not result from the hardware and is not bound to a particular redundant hardware architecture.

Software architectures and structural approaches independent of the particular problem solved or function that is achieved. As a consequence, documents describing such techniques for a particular purpose or hardware architecture as covered by groups G06F 11/14 and G06F 11/16 should be classified here as well.

Examples:

- fault-tolerance using data-diversity (e.g. by using different equivalent input data sets for each retry of a function), corrective actions, e.g. following a plausibility check;
- measures taken before run time (e.g. duplication of instructions for comparison at compile time) or robust data structures.

Insert: The following new Relationships section.

Relationships with other classification places

Documents where the kind of fault tolerance used (active-active, voting, active-passive) is fixed by the hardware architecture and cannot be influenced by the software are classified in group G06F 11/16.

G06F11/1482

Replace: The existing Definition statement with the following updated statement.

Definition statement

This place covers:

Software layers (on top of the operating system or integrated in the operating system) which make applications which are not fault-tolerant run in a fault-tolerant way. Typically, this is done by scheduling requests of the application more than once.

Examples:

- fault-tolerant cluster software;
- operating systems detecting a faulty process and creating a further copy of the same process on the same processor (but potentially in another memory area).

Replace: The existing Special rules text with the following updated text.

Special rules of classification

If the software layer implements an application restart or rejuvenation mechanism, this feature is to be additionally classified in G06F 11/1438.

Documents where there is necessarily some hardware redundancy should be classified using indexing codes, as well as in the HW redundancy groups.

Documents which describe said middleware/OS techniques in combination with only one redundant hardware architecture should always be classified into G06F 11/16. If a middleware/OS technique is explicitly suitable for multiple redundant hardware architectures, it should be classified in group G06F 11/1482 in addition to group G06F 11/16.

Restart related aspects of the second example should be classified additionally in G06F 11/1438.

Note that G06F 11/1482 as such does not imply failover mechanisms, although the underlying hardware becomes redundant because of the software layer. Hence, this hardware redundancy aspect requires classification as well in group G06F 11/2023.

G06F11/1487

Replace: The existing Definition statement with the following updated statement.

Definition statement

This place covers:

Error detection wherein the output of multiple versions (typically based on different source codes) of the application code or portions thereof are compared or voted. The multiple versions can be in different programming languages, different compilers or implementation of alternative algorithms. The versions may be executed sequentially, concurrently or in parallel on different hardware (thereby making the latter redundant).

G06F11/1489

Replace: The existing Definition statement with the following updated statement.

Definition statement

This place covers:

Error detection wherein an acceptance test (usually a plausibility check or a limit on execution time) is performed on critical code blocks. If the test is not passed, the output of another execution of the same or an alternative version of the block is used for recovery. The executions may be sequential or parallel. Note that an acceptance test is performed using the output of only one of the executions.

Replace: The existing Special rules text with the following updated text.

Special rules of classification

If the other execution is systematically performed on the same hardware, additional classification in group G06F 11/1497 should be considered.

G06F11/1492

Definition statement

Replace: The existing Definition statement with the following updated statement.

This place covers:

Software performing run-time replications wherein the redundancy is inherent in the application itself. Thus, the application does not rely on any other layer to be fault tolerant. The redundant portions are necessarily identical, since otherwise the redundancy is not realised at runtime, but is hard coded.

G06F11/1497

Replace: The existing Definition statement with the following updated statement.

Definition statement

This place covers:

Pieces of software, e.g. modules, functions or complete applications, always being executed two or more times sequentially or concurrently (e.g. as threads) on a single processing unit in order to address transient faults.

Aspects relating to the provision of the identical input to all executions of the software. When three or more executions are foreseen, this can be used for error correction (not only for error detection).

Techniques used to instantiate multiple executions of redundant software, the temporary storage of intermediary results or the duplication of contexts for each instance, as well as the measures taken for the subsequent error detection or fault masking.

(Non-redundant) hardware support for time redundant execution.