Internal distribution code:

(A) [-] Publication in OJ
(B) [-] To Chairmen and Members
(C) [-] To Chairmen
(D) [X] No distribution

Datasheet for the decision of 19 December 2025

Case Number: T 0762/24 - 3.5.07

Application Number: 18800422.0

Publication Number: 3701392

IPC: G06F16/26, G06F8/10, G06F8/34

Language of the proceedings: EN

Title of invention:
TRANSFORMING A SPECIFICATION INTO A PERSISTENT COMPUTER PROGRAM

Applicant:
AB Initio Technology LLC

Headword:
Specification transformation/AB INITIO TECHNOLOGY

Relevant legal provisions:
EPC Art. 56, 84, 111(1), 123(2)
RPBA 2020 Art. 11, 13(1), 13(2)

Keyword:

Claim 1 of the main request - added subject-matter (no)
Claim 1 of the main request - clarity (yes)
Claim 1 of the main request - inventive step (yes)
Claim 1 of the main request - remittal - special reasons for remittal (yes)

Europäisches Patentamt
European Patent Office
Office européen des brevets

Beschwerdekammern

Boards of Appeal

Chambres de recours

Boards of Appeal of the
European Patent Office
Richard-Reitzner-Allee 8
85540 Haar
GERMANY
Tel. +49 (0)89 2399-0

Case Number: T 0762/24 - 3.5.07

D E C I S I O N
of Technical Board of Appeal 3.5.07
of 19 December 2025

Appellant: (Applicant)	AB Initio Technology LLC 201 Spring Street Lexington, Massachusetts 02421 (US)
Representative:	Herrmann, Daniel Boehmert & Boehmert Anwaltspartnerschaft mbB Pettenkoferstrasse 22 80336 München (DE)
Decision under appeal:	**Decision of the Examining Division of the European Patent Office posted on 6 December 2023 refusing European patent application No. 18800422.0 pursuant to Article 97(2) EPC.**

Composition of the Board:

Chair	R. de Man
Members:	C. Barel-Faucheux
	E. Mille

Summary of Facts and Submissions

I. The appellant (applicant) appealed against the examining division's decision refusing European patent application No. 18 800 422.0, published under international application number WO 2019/084172 A1.

II. The examining division cited the following documents:

D1: WO 2005/043388 A1, 12 May 2005

D2: US 2016/0132360 A1, 12 May 2016

D3: WO 98/00791 A1, 8 January 1998

III. The examining division decided that the subject-matter of claims 1 to 15 of the main request, of the independent claims of the first and second auxiliary requests, of claim 1 of the third auxiliary request, and of the independent claims of the fourth and fifth auxiliary requests did not involve an inventive step within the meaning of Article 56 EPC with regard to the disclosure of document D2, in combination with the disclosure of document D1 for some of the dependent claims.

IV. With the statement of grounds of appeal, the appellant requested that the examining division's decision be set aside and that a European patent be granted based on the main request or one of the first to fifth auxiliary requests considered in the contested decision and resubmitted with the statement of grounds of appeal.

V. In a communication accompanying a summons to oral proceedings, the board expressed its preliminary opinion that the subject-matter of claim 1 of the main request and of the fifth auxiliary request lacked clarity (Article 84 EPC), that the subject-matter of claim 1 of the main request did not meet the requirements of Article 123(2) EPC or those of Article 56 EPC, that the subject-matter of claim 1 of the first to fourth auxiliary requests did not meet the requirements of Article 56 EPC, and that the subject-matter of claim 1 of the fourth auxiliary request did not meet the requirement of Article 123(2) EPC. It stated that, during the oral proceedings, it might have to be discussed whether the subject-matter of claim 1 of the fifth auxiliary request involved an inventive step (Article 56 EPC).

VI. With its letter of reply, the appellant submitted new auxiliary requests I, III, V, VII, IX, XI and XII and renumbered its pending auxiliary requests as auxiliary requests II, IV, VI, VIII and X. The appellant also provided further arguments.

VII. Oral proceedings were held as scheduled on 19 December 2025. During the oral proceedings the appellant submitted a new set of claims as auxiliary request XIII, reordered auxiliary request XIII as the main request and renumbered the other requests on file (i.e. main request and auxiliary requests I to XII) as auxiliary requests I to XIII, respectively. At the end of the oral proceedings, the Chair announced the board's decision.

VIII. The appellant's final requests were that the decision under appeal be set aside and that a patent be granted

on the basis of the claims of the main request or one of auxiliary requests I to XIII.

IX. Independent claim 1 of the main request reads as follows (itemisation by the board):

"F1 A method performed by a computer system in transforming a specification into a computer program that processes one or more values of one or more fields in a structured data item, including:

F2 accessing a specification (19b, 44, 86) that specifies a plurality of modules to be implemented by the computer program (14, 18) for processing the one or more values of the one or more fields in the structured data item;

F3 transforming the specification (19b, 44, 86) into the computer program (14, 18) that implements the plurality of modules, wherein the transforming includes:

for each of one or more first modules of the plurality of modules:

F31 identifying one or more second modules of the plurality of modules that each receive input that is at least partly based on an output of the first module; and

F32 determining, before execution of the computer program for data processing, an output data format of the first module such that the first module is configured to output only one or more values of one or more fields of the structured data item that are each (i) accessible to the first module, and (ii) specified, in an input data format of at least one of the one or more second modules, as input required by the at least one second module, the input being required to fulfill data processing requirements of the at least one second module,

F321 wherein the output data format of the first module is determined such that a version of the structured data item to be outputted by the first module for further processing by the at least one second module when the computer program is executed includes all of the one or more fields of the structured data item, except for those one or more fields that are not specified as input into any one of the one or more second modules, and wherein each first module is configured with the output data format by

F322 executing, before executing the computer program (14, 18), first, second and third passes for each particular module of the plurality of modules to determine and save the output data format for the particular module specifying which fields are required by modules downstream to the particular module and to determine and save an input data format for the particular module specifying which fields are accessible to the particular module, the executing of the passes including:

F323 during the first pass, the particular module broadcasts, to modules upstream to the particular module, one or more messages that include data representing those one or more fields that are required by itself and by any modules downstream of the particular module, wherein the particular module that performs the broadcast is a broadcasting module and wherein the upstream modules that receive the broadcast are recipient modules,

F324 during the second pass, the recipient modules transmit to the broadcasting module one or more messages specifying which recipient modules can provide values of the required fields, and

F325 during the third pass, the broadcasting module analyzes the messages received from the recipient modules and, in response, the broadcasting module transmits back to the recipient modules one or more messages specifying which recipient module is responsible for transmitting which field to the broadcasting module;

F33 saving, in persistent memory, the computer program (14, 18), with the saved computer program (14, 18) specifying the output data format for each of the one or more first modules;

F34 receiving, in a data stream, the structured data item including one or more data records; and

F35 for at least one of the one or more data records,

executing the computer program (14, 18) to process the at least one of the one or more data records; and

F36 during execution of at least one of the one or more first modules of the saved computer program, looking up from persistent storage the output data format of the at least one of the one or more first modules of the saved computer program and outputting, by the at least one of the one or more first modules, a version of the at least one of the one or more data records that includes all of one or more fields of the at least one of the one or more data records, except for those one or more fields in the at least one of the one or more data records that are not specified in the looked-up output data format as input into any one of the one or more second modules that depend at least partly on an output of the at least one of the one or more first modules."

Independent claim 12 of the main request reads as follows:
"A computer system for transforming a specification (19b, 44, 86) into a computer program (14, 18) that processes one or more values of one or more fields in a structured data item, the computer system including:
one or more processing devices; and
one or more machine-readable hardware storage devices storing instructions that are executable by the one or more processing devices to perform the operations of the method of any one of claims 1 to 11."

Independent claim 13 of the main request reads as follows:
"One or more machine-readable hardware storage devices for transforming a specification into a computer program (14, 18) that processes one or more values of one or more fields in a structured data item, the one

or more machine-readable hardware storage devices storing instructions that are executable by one or more processing devices to perform the operations of the method of any one of claims 1 to 11."

X. In view of the outcome of this decision, it is not necessary to reproduce the text of the auxiliary requests.

Reasons for the Decision

The application

1. The application relates to a method performed by a computer system for transforming a "specification" into a computer program that processes one or more values of one or more fields in a structured data item.

Admission of the main request

2. During the oral proceedings, the board informed the appellant that it was of the opinion that the subject-matter of claim 1 of the fifth auxiliary request considered in the contested decision (and corresponding, at the beginning of the oral proceedings, to auxiliary request X) involved an inventive step but that claim 1 of this request should be reworded in view of the objections under Article 84 EPC raised by the board for the first time in its communication. Claim 1 of the present main request is based on claim 1 of the fifth auxiliary request considered in the contested decision with amendments made during the oral proceedings to meet the requirements of Article 84 EPC. The board therefore exercises its discretion to admit the main request into the appeal proceedings (Article 13(1) and (2) RPBA).

Basis of claim 1 of the main request

3. Features F1 to F3 and F31 of claim 1 of the main request correspond to the beginning of claim 1 as originally filed (before the "formatting" step in claim 1 as originally filed). The "saving" step stipulated in feature F33 corresponds to the saving step in claim 1 as originally filed, except that it no longer specifies that the output data format is "formatted" (see point 4. below).

4. Feature F32 corresponds to the "formatting" step in claim 1 as originally filed; however, since the expression "formatting an (output) data format" was not considered to be clear by the board, the appellant replaced this expression with "determining [...] an (output) data format", which is an alternative expression used on page 53, lines 4 to 17 of the application as published. Moreover, this determining step is performed "before execution of the computer program for data processing" (see page 42, lines 19 to 25).

5. Features F322 to F325 correspond to claim 11 as originally filed (which is dependent on claim 1 as originally filed) with minor editorial amendments (i.e. without the specification in feature F322 that it is the "formatted" output data format which is saved, and with the information added that it is the execution of the "passes" which includes steps F323 to F325).

6. Features F34 and F35 correspond to the beginning of claim 12 as originally filed (which is dependent on claim 1 as originally filed) before the "based a [sic] saved output data format [...] at least partly based on

the output of the first module" (i.e. the end of claim 12 as originally filed).

7. As for feature F36, the passage on page 45, lines 5 to 14 of the description as originally filed reads as follows (emphasis added by the board):
"Then, during execution of the dataflow graph, the data processing system queries for this stored data, during execution of an entity. For example, upon execution of a particular entity, the data processing system looks up, in the database, the data for that entity (e.g., based on a unique identifier for the entity). [...] the looked-up data specifies which fields are required by downstream entities. In this example, the data processing system is configured to drop those fields (e.g., by deleting them from storage or by preventing storage of data for those fields) that are not required by downstream entities - e.g., upon completion of data processing by a current entity (e.g., an entity being currently executed and associated with stored data specifying which fields are and are not required by downstream entities).".

Moreover, the passage on page 47, lines 2 to 7 of the description as originally filed reads as follows (emphasis added by the board):
"Now, when the dataflow graph shown in FIG. 5 is executed (in real-time) by data processing system 4 (FIG. 1 A), the data processing system 4 (executing the dataflow graph) can determine to drop field A, following processing by entity 130, e.g., based on a data look-up in memory (or a data repository) of the data specifying to drop field A upon completion of data processing by entity 130.".

The board notes that a "dataflow graph" is a program (or a representation of a program).

This is what feature F36 stipulates, but in slightly different wording:

- first, feature F33 stipulates that the computer program saved in persistent memory specifies the output data format for each of the one or more first modules;
- then, as expressed in feature F36, during execution of at least one of those modules, the output data format of the at least one of the one or more first modules is looked up from persistent storage [i.e. memory];
- a version of the at least one of the one or more data records that includes all of one or more fields of the at least one of the one or more data records, except for those one or more fields in the at least one of the one or more data records that are not specified in the looked-up output data format as input into any one of the one or more second modules (that depend at least partly on an output of the at least one of the one or more first modules), is outputted by the at least one of the one or more first modules.

8. Feature F321 is the resulting output data format of feature F326, except that feature F321 uses the terminology "one or more fields" of the "structured data item" instead of "one or more fields" of the "at least one of the one or more data records". This is because feature F34 stipulates receiving, in a data stream, "the structured data item including one or more data records".

9. Therefore, claim 1 of the main request fulfils the requirements of Article 123(2) EPC.

Support in the description of claim 1 of the main request

10. Claim 1 of the main request is based on the embodiment illustrated in Figure 5 of the application as originally filed, as is reproduced below, together with the description as originally filed, from page 45, line 15 to page 49, line 20.

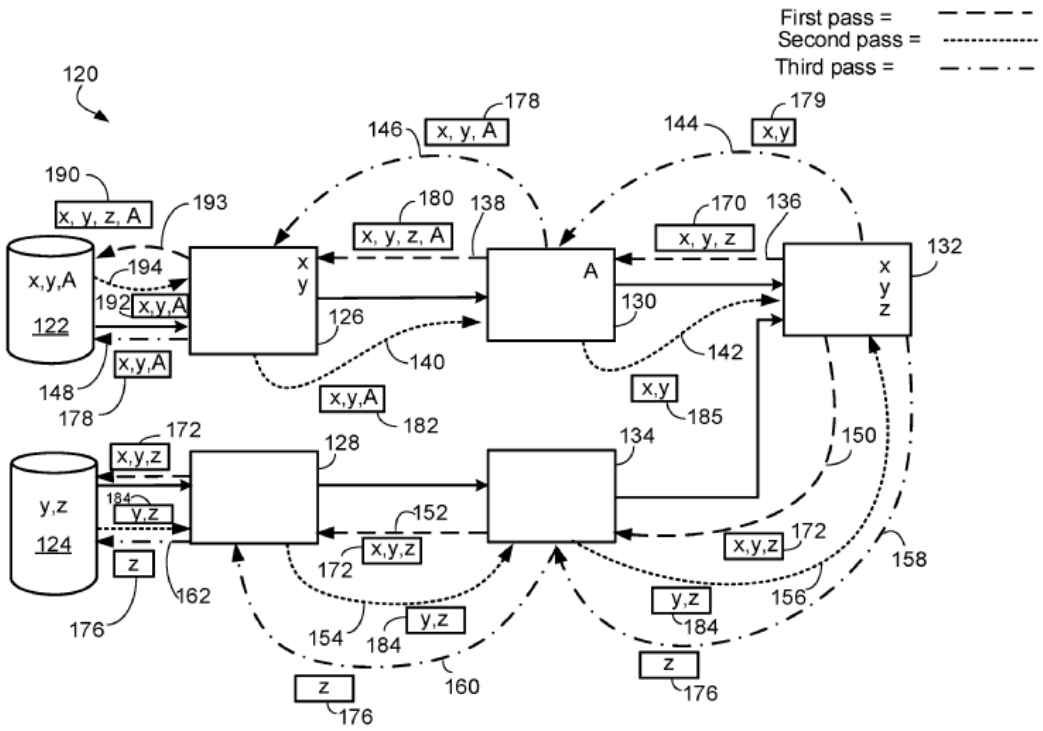


FIG. 5

10.1 In the example illustrated by Figure 5, the entity (or "module" in claim 1) 132 requires three fields as the input: x, y and z. The entity 130 requires one field: A. The entity 126 requires two fields: x and y. The data source 122 stores data records with fields x, y and A (corresponding to a "structured data item" having "one or more fields" of feature F1 of claim 1).

10.2 To track which fields are required by which entities/ modules and when the data processing system can drop a field (see feature F36: ", except for those one or more fields in the at least one of the one or more data records that are not specified in the looked-up output data format as input into any one of the one or more second modules"), the data processing system performs the following techniques as part of the configuration of a dataflow graph and prior to data processing by the dataflow graph (see feature F32, "before execution of the computer program for data processing" and feature F322, "before executing the computer program").

10.2.1 First, during the first pass (corresponding to the first pass of feature F323 of claim 1), the entity 132 sends the message 170 (including data specifying that fields x, y and z are required by the entity 132) to the entity 130, as shown by the arrow 136. In turn, the entity 130 receives the message 170 and adds to it any fields that are required by the entity 130. It then transmits the message 180 (i.e. data specifying that fields x, y, z and A are required) to the entity 126, as shown by the arrow 138. In turn, the entity 126 sends the message 190 (specifying that fields x, y, z and A are required by itself or by one or more downstream entities) to the data source 122, as shown by the arrow 193.

10.2.2 Because the data source 122 stores values for fields x, y and A, it transmits the message 192 to the entity 126, as shown by the arrow 194 during a second pass (corresponding to the second pass of feature F324 of claim 1). The message 192 specifies that the data source 122 stores or otherwise has access to values for fields x, y and A. In turn, the entity 126 transmits

the message 182 (with fields x, y and A) to the entity 130, as shown by the arrow 140. The entity 130 receives the message 182 and identifies which, if any, of the fields specified in the message 182 are required by downstream entities. As shown by the arrow 142, it then transmits, to the entity 132, the message 185 with data indicating that the entity 130 can provide fields x and y.

10.2.3 The entity 130 also stores, in a memory or data repository, data specifying that field A can be dropped or otherwise removed from storage once field A is processed by the entity 130 (see feature F36 of claim 1 and point 7. above).

10.2.4 In parallel with (or subsequently to) transmitting the message 170, the entity 132 also transmits the message 172 to the entity 134, during the first pass as shown by the arrow 150. In this example, the message 170 is a the same message as the message 172. Since the entity 134 does not require any fields of its own, it forwards the message 172 to the entity 128 as shown by the arrow 152. The entity 128 forwards the message 172 to the source 124 (still during the first pass), which in turn responds with the message 184 (during the second pass), specifying that the source 124 can provide fields y and z.

10.2.5 In turn, during the second pass, the entity 128 responds with the message 184 specifying that the entity 128 can provide values for fields y and z. The entity 128 transmits the message 184 to the entity 134, as shown by the arrow 154 during the second pass. The entity 134 forwards the message 184 to the entity 132, as shown by the arrow 156.

10.2.6 In this example, the entity 132 has received two messages (i.e. the messages 184 and 185), each of which specify which fields required by the entity 132 can be provided by upstream entities. In particular, the message 185 specifies that fields x and y can be provided by the entity 130 and the message 184 specifies that fields y and z can be provided by the entity 134. Since the entity 132 can receive a value for field y from two different entities, the entity 132 selects one of the entities from which to make the request and to receive values for field y, for example randomly. In the example shown in Figure 5, the entity 132 selects the entity 130 for receiving field y.

10.2.7 During a third pass (see feature F325), the entity 132 transmits the message 179 to the entity 130, as shown by the arrow 144. In this example, the message 179 specifies that the entity 132 chooses to receive fields x and y from the entity 130. In turn, the entity 130 generates the message 178 that specifies that the entity 130 will receive values of fields x, y and A from the entity 126. The message 178 is transmitted from the entity 130 to the entity 126, as shown by the arrow 146. During the third pass, the entity 126 transmits the message 178 to the data source 122, as shown by the arrow 148.

10.2.8 The data source 122 compares the content of the message 178 with fields it stores or otherwise accesses to determine which fields, if any, the data source 122 can drop and delete from storage. In this example, the message 178 indicates that all the fields of the data source 122 are required by downstream entities. The data source 122 is therefore not configured to drop any fields.

10.2.9 The entity 132 transmits the message 176 to the entity 134 specifying that the entity 132 will receive field z from the entity 134, as shown by the arrow 158, during the third pass. In turn, the entity 134 is configured by the data processing system to transmit a value of field z to the entity 132. The entity 134 also transmits the message 176 to the entity 128, as shown by the arrow 160, during the third pass. As such, the entity 128 is also configured to transmit values for field z to the entity 134. In turn, the entity 128 forwards the message 176 to the data source 124, as shown by the arrow 162, during the third pass.

10.2.10 The data source 124 compares the contents of the message 176 (i.e., specifying field z) with fields accessed by the data source 124, which in this example are fields y and z. Based on the comparison, the data processing system determines that field y is not included in the message 176. As such, the data processing system determines that no downstream entities are relying on the data source 124 for values of field y. As such, the data source 124 is configured to drop field y, e.g., upon execution of the dataflow graph shown in Figure 5.

11. Therefore, claim 1 of the main request is supported by the description (Article 84 EPC).

Clarity of claim 1 of the main request

12. The clarity objections to claim 1 of the then main request raised by the board in its communication do not apply to the wording of the present claim 1 and therefore do not need to be discussed any further.

13. The board additionally raised a clarity objection to the features of claim 1 of the then fifth auxiliary request, which correspond to features F322, F323, F324 and F325 of the present claim 1.

13.1 According to features F323, F324 and F325, during the first, second and third passes, various modules "broadcast", "transmit" and "analyze" messages, which means that the modules execute these steps; however, feature F322 states that the three passes are executed "before executing the computer program" which implements the modules (see feature F2).

13.2 The board now accepts that feature F322 does not contradict features F323, F324 and F325. Instead, these four features taken together imply that, in addition to their module-specific functionality as part of the "computer program", each module implements generic functionality for "broadcasting", "transmitting" and "analyzing" messages in order to configure the "output formats".

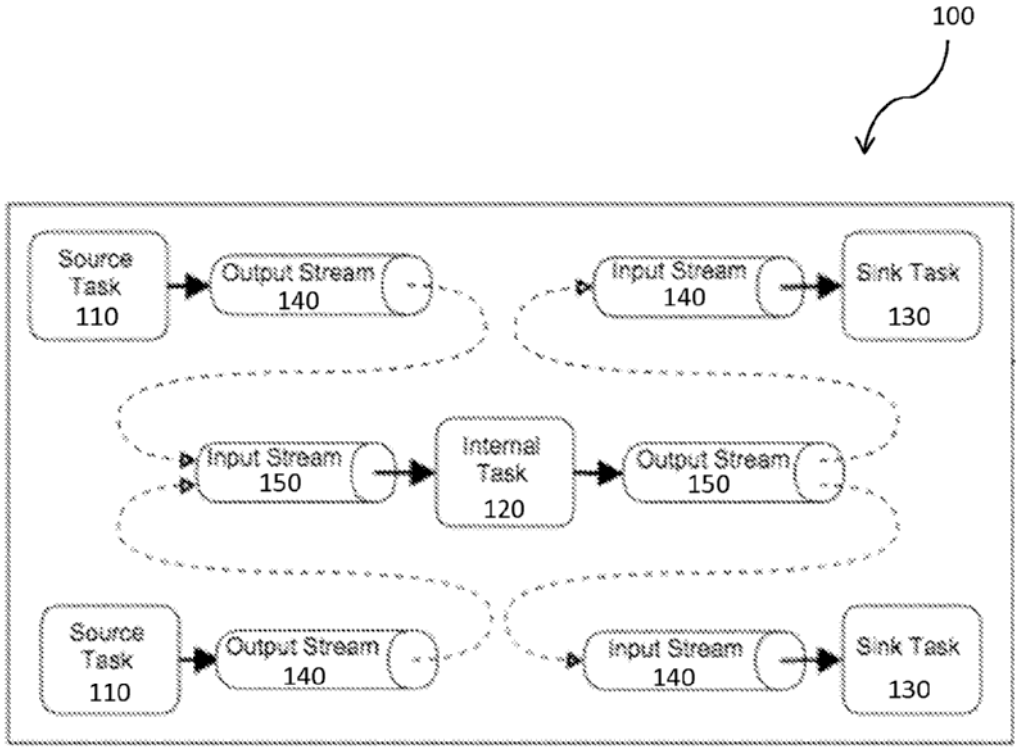
13.3 The board therefore considers that claim 1 complies with the requirements of Article 84 EPC.

Inventive step of claim 1 of the main request - documents D1 to D3

14. The board agrees with the appellant and the examining division that **document D2** is the prior art closest to the method of claim 1.

14.1 The method disclosed in document D2 relates to the configuration of stream schemas for the exchange of data objects used in distributed processing systems (paragraph [0002]).

14.2 In a distributed stream-processing environment, process topologies consist of a set of tasks that are interconnected by streams that move tuples from their upstream tasks to their downstream tasks. Tasks and task workers can be computer programs or executable code modules that can manipulate tuples or data objects, or transmit/receive tuples or data objects to or from one or more streams. During processing, tasks consume tuples from their input streams and emit tuples to their output streams, where each stream has its own stream schema that defines the fields of the stream's tuples (paragraph [0034]). Figure 1 of document D2, as reproduced below, shows a distributed process 100 with tasks and interconnecting streams. Tasks can be any operation, procedure, program flow step, etc. that operates on the tuples. A distributed process 100 can include multiple types of task, e.g. source tasks 110, internal tasks 120, and sink tasks 130 that perform operations on tuples (paragraph [0035]).



14.3 In particular, document D2 discloses a method, performed by a computer system, in which a "task worker" running on a worker server receives, over a network, a process specification specifying a task to be executed by the task worker. The executed task includes generating an output data object (for an output data stream) based in part on an input data object (from an input data stream) (paragraph [0005]; see also paragraph [0019]).

14.4 The process specification is accessed to specify the required fields to be read for executing the task and to specify the generated fields in the input or output data object that will be written to during or subsequent to the execution of the task (paragraphs [0006] and [0007]).

14.5 The task worker executes the task and generates the output data object. The task includes reading the

required fields from the input data object and writing to the generated fields in the output data object (paragraph [0008]).

14.6 The process specification specifies a task to be executed by the task worker and includes an "iteration configuration" that specifies required fields in the input data object in order for the task worker to execute the task. In particular, the iteration configuration is used by the task to process input tuples and fields and/or generate output tuples and fields. Also, the process specification specifies generated fields in the output data object to be written to during execution of the task (paragraphs [0009], [0010] and [0040]).

14.7 An "internal" task 120 can have a task execution method 510 and a task framework 520 that can be used by the iteration configuration 420 to create the required tuple fields 530, prior to the tuple 220 being processed by the internal task 120. The iteration configuration 420 can also create the generated tuple fields 540, after the tuple 220 has been processed by the internal task 120 (paragraph [0041]).

14.8 In addition, the process specification can include a "task configuration" having: the iteration configuration, task designations providing an indication of which of the tasks are to be executed, and task parameters specifying parameters required for the tasks (paragraph [0014]).

14.9 The board notes that the "data object" in the method of document D2 can be equated to the "structured data item" in claim 1 of the main request, while the "tasks to be executed" in the method in document D2 can be

equated to the "plurality of modules to be implemented by the computer program" in claim 1 of the main request.

14.10 As mentioned in point 14.2 above, "stream schemas" define the fields of the tuples transmitted in a stream from a first module to a second module (paragraph [0034]), i.e. they are "output data formats" within the meaning of claim 1. Adding a generated field 435 to the iteration configuration 420 can add the field to all downstream stream schemas. Similarly, filtering out of fields by the stream tuple field filter 720 can remove the field from all downstream schemas. The user-modifiable field filter column 640 specifies whether a field in the input data object is to be retained when generating the output data object. The field filter column 640 can provide an indication of whether to either keep or drop the tuple field 260 from the tuple 220 before the tuple 220 is delivered by the stream 210 to the downstream task 240. In the method in document D2, as tasks are selected and streams connecting the tasks are defined, some of the required fields 430 and generated fields 435 can be automatically established by the specification editor 750 (or another connected process)(paragraphs [0003], [0018], [0042], [0044] and [0045]; claim 5).

14.11 Hence, document D2 suggests removing fields from stream schemas. In the board's view, the skilled person would consider removing all unnecessary fields, i.e. fields which are not used downstream, in order to minimise the amount of data being transmitted, thus freeing up computing resources.

14.12 While document D2 does disclose a process for resolving output data formats in the form of stream schemas by

performing a breadth-first traversal of the data flow topology graph using a priority task queue (see Figure 8 and paragraphs [0047] and [0048]), it does not disclose the specific process of determining minimal output data formats specified in claim 1. The distinguishing features of claim 1 of the main request over the disclosure of document D2 are therefore at least features F322 to F325, i.e. the execution of the first to third passes.

14.13 In particular, the "tasks to be executed" or modules in the method in document D2 do not broadcast messages to, or receive messages from, the other tasks or modules upstream or downstream of the particular task or module as stipulated in claim 1 of the main request, i.e.:

- messages that include data representing those one or more fields that are required by the broadcasting module and by any modules downstream of the particular module (feature F323);
- messages specifying which recipient modules can provide values of the required fields (feature F324);
- messages specifying which recipient module is responsible for transmitting which field to the broadcasting module (feature F325).

14.14 These distinguishing features provide an alternative, decentralised and parallel implementation of a process for determining minimal output data formats. The board is not convinced that, using only their common general knowledge, the skilled person is capable of arriving at the claimed solution, for which document D2 does not provide a pointer and which relies on specific adaptations of the modules to provide generic functionality for "broadcasting", "transmitting" and

"analyzing" messages separate from their module-specific functionality (see point 13.2 above).

14.15 The system in **document D1** includes a plurality of map components where each map component has one or more ports for accepting data and for producing data and each map component encapsulates a particular dataflow pattern. Compiler tools for organising and linking the map components using the ports into a dataflow application are provided, as well as an executor for creating and managing data communication among map components in the dataflow application and executing the dataflow application with data supplied to the system (page 4, first paragraph). It does not disclose features F322 to F325, i.e. the execution of first to third passes.

14.16 **Document D3** discloses a method and apparatus by which a graph can be used to invoke computations directly. The invention provides methods for getting information into and out of individual processes represented on a graph, for moving information between the processes, and for defining a running order for the processes (page 3, first paragraph). It does not disclose features F322 to F325, i.e. the execution of first to third passes.

14.17 Since neither the common general knowledge nor the cited prior art suggests the claimed solution, the board concludes that the subject-matter of claim 1 and of the corresponding independent claims 12 and 13 involves an inventive step (Article 56 EPC).

15. *Remittal for further prosecution*

15.1 In view of the above, the decision under appeal is to be set aside; however, the board has not examined

whether the amendments made to the independent claims require adaptations to be made to the dependent claims. Moreover, the description and drawings may still need to be adapted. These are special reasons for remitting the case to the examining division for further prosecution on the basis of the main request (Article 111(1) EPC and Article 11 RPBA).

Order

For these reasons it is decided that:

1. The decision under appeal is set aside.

2. The case is remitted to the examining division for further prosecution.

The Registrar: The Chair:

S. Lichtenvort R. de Man

Decision electronically authenticated