BESCHWERDEKAMMERN DES EUROPÄISCHEN PATENTAMTS	**BOARDS OF APPEAL OF THE EUROPEAN PATENT OFFICE**	**CHAMBRES DE RECOURS DE L'OFFICE EUROPÉEN DES BREVETS**

Internal distribution code:

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

Datasheet for the decision
of 17 December 2020

Case Number:	T 1190/16 - 3.4.03
Application Number:	10197199.2
Publication Number:	2472450
IPC:	G06Q10/00
Language of the proceedings:	EN

Title of invention:
A search method for a containment-aware discovery service

Applicant:
Hasso-Plattner-Institut für
Softwaresystemtechnik GmbH

Headword:

Relevant legal provisions:
EPC Art. 56

Keyword:
Inventive step - problem and solution approach - ex post facto analysis - main request (no) - auxiliary request (no)

This datasheet is not part of the Decisior
It can be changed at any time and without notic

Decisions cited:

T 0914/02
German Federal Court (BGH) X ZR 173/07 (Walzgerüst II)

Catchword:

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
Fax +49 (0)89 2399-4465

Case Number: T 1190/16 - 3.4.03

DECISION
of Technical Board of Appeal 3.4.03
of 17 December 2020

Appellant:
(Applicant)

Hasso-Plattner-Institut für
Softwaresystemtechnik GmbH
Prof.-Dr.-Helmert-Str. 2-3
14482 Potsdam (DE)

Representative:

Molnia, David
Df-mp Dörries Frank-Molnia & Pohlman
Patentanwälte Rechtsanwälte PartG mbB
Theatinerstrasse 16
80333 München (DE)

Decision under appeal:

Decision of the Examining Division of the European Patent Office posted on 26 February 2016 refusing European patent application No. 10197199.2 pursuant to Article 97(2) EPC.

Composition of the Board:

Chairman G. Eliasson
Members: A. Böhm-Pélissier
G. Decker

Summary of Facts and Submissions

I. The appeal is against the decision of the Examining Division to refuse European patent application No. 10 197 199. The refusal was based on the ground of lack of inventive step (Article 56 EPC).

II. **Reference** is made to the following documents:

D5= UNIVERSITY OF CAMBRIDGE ET AL:
"High level design for Discovery Services",
INTERNET CITATION, 15 August 2007, pages 1-112,
XP002638393, Retrieved from the Internet:
URL:http://www.bridge-project.eu/data/File/BRIDGE%20WP02%20High%20level%20design%20Discovery%20Services.pdf, [retrieved on 2011-05-23]

D7= MENDLING J ET AL:
"Process Mining of RFID-Based Supply Chains",
COMMERCE AND ENTERPRISE COMPUTING,
2009 CEC '09 IEEE CONFERENCE, IEEE, PISCATAWAY,
NJ, USA, 20 July 2009, pages 285-292,
XP031516536, ISBN: 978-0-7695-3755-9

D8= GYEONGTAEK LEE ET AL:
"Discovery Architecture for the Tracing of
Products in the EPCglobal Network",
EMBEDDED AND UBIQUITOUS COMPUTING,
2008 EUC '08. IEEE/IFIP INTERNATIONAL
CONFERENCE, IEEE, PISCATAWAY, NJ, USA,
17 December 2008, pages 553-558, XP031408685,
ISBN: 978-0-7695-3492-3

III. At the oral proceedings before the Board on

17 December 2020, the Appellant (Applicant) **requested** that the decision under appeal be set aside and that a patent be granted on the basis of the Main Request or the Auxiliary Request filed with the letter dated 16 December 2020.

IV. **Claim 1** of the Main Request reads (Board's Labelling):

(A') A search method for identifying from an original set of event notifications stored in a discovery service database a subset of event notifications comprising event notifications relevant for a query item,

(B') wherein the method is carried out by an application logic of a system for running a discovery service,

(C') the system comprising the discovery service database,

(D') wherein each event notification is either an object event notification or an aggregation event notification,

(E') wherein each aggregation event notification comprises a timestamp,

(F') wherein each object event notification comprises a timestamp and an object identifier, the method being characterized in that

(G') each aggregation event notification further comprises an action, a parent identifier, and a child identifier;

(H') and in that determining the subset comprises the following steps:

(I') (A) adding to the subset each object event notification having an input identifier as object identifier and a timestamp from an input time span;

(J') (B) adding to the subset each aggregation event notification having the input identifier as a child identifier and a timestamp from the input time span;

(K') (C) invoking steps (A) to (C) for each aggregation event notification that was added in the last step (B) and has addition as an action, using the parent identifier of the corresponding aggregation event notification as the input identifier,
(L') and using the time span limited by the timestamps of the first and the last aggregation event notification added in the last step (B) as the input time span.

Claim 1 of the **Auxiliary Request** reads (Board's Labelling, striking-through for deletions, underlining for additions with respect to the Main Request):
(A'') A ~~search~~-method for identifying from an original set of event notifications stored in a discovery service database a filtered set of ~~a subset of event notifications comprising~~ event notifications relevant for a query item,
(B'') wherein the method is carried out by an application logic of a system for running a discovery service,
(C'') the system comprising the discovery service database,
(D'') wherein each event notification is either an object event notification or an aggregation event notification,
(E'') wherein each aggregation event notification comprises a timestamp, a parent identifier and a child identifier,
(F'') wherein each object event notification comprises a timestamp and an object identifier, the method being characterized in that
(G'') each aggregation event notification further comprises an action, a parent identifier, and a child identifier; and in that

(H'') ~~and wherein~~ determining the ~~subset~~ filtered set comprises the following steps:

performing a search method to obtain a subset of event notifications comprising event notifications relevant for a query item, the search method comprising the following steps:

(I'') (A) adding to the subset each object event notification having an input identifier as object identifier and a timestamp from an input time span;

(J'') (B) adding to the subset each aggregation event notification having the input identifier as a child identifier and a timestamp from the input time span;

(K'') (C) invoking steps (A) to (C) for each aggregation event notification that was added in the last step (B) and has addition as an action, using the parent identifier of the corresponding aggregation event notification as the input identifier,

(L'') and using the time span limited by the timestamps of the first and the last aggregation event notification added in the last step (B) as the input time span

(M'') performing a filter method in order to identify from the obtained subset of event notifications those event notifications that are relevant for a query item, the filter method comprising the following steps:

(N'') creating a filtered list as the filtered set, which is initially empty;

(O'') creating a stack, which is initially empty;

pushing the unique identifier of the query item onto the stack; and

repeating the following steps for each event notification of the set in chronological order:

(P'') pushing the parent identifier of the current event notification on the stack and adding the current event notification to the filtered list if the current event notification is an aggregation event, and comprises addition as an action, and comprises the topmost unique identifier of the stack as a child identifier;

(Q'') removing the topmost unique identifier from the stack and adding the current event to the filtered list if the current event notification fulfills predetermined conditions, wherein the predetermined conditions comprise: the current event notification is an aggregation event, and comprises deletion as an action, and comprises the topmost unique identifier of the stack as a parent identifier; and

(R'') adding the current event notification to the filtered events list if the current event notification is an object event notification and its object identifier is stored somewhere in the stack.

V. The Appellant argued essentially as follows:
Document D8 neither alone nor in combination with document D7 or the common general knowledge taught

(i) to operate on event notifications, which are stored in the discovery service and comprise action and aggregation information, rather than on events, which are stored in on-site repositories;

(ii) running an iterative discovery service algorithm on a system comprising said event notifications by narrowing the data space during each recursion by way of an input time span;

(iii) applying a two step approach with a pre-search as claimed in the Main Request and a second filter algorithm as specified in the Auxiliary Request.

Reasons for the Decision

1. The invention as claimed

1.1 The invention concerns a "discovery service" which is suitable for tracking and tracing a "query item" represented by a "unique identifier" in a "unique identifier network". An example for such a network is a network deployed in a supply chain spanning over one or more companies. These networks rely on unique identifier technologies, such as tags (radio frequency identification [RFID] tags) and unique identifier coding schemes (bar-codes), and allow item tracking, item tracing, item authentication, or item supply chain analysis.

1.2 Examples for unique identifier coding schemes are the Electronic Product Code ("EPC") coding scheme and the Unique Consignment Reference ("UCR") coding scheme. Every time a unique identifier is read, a piece of data is generated ("object event"). Each event may be stored on an event server, which is typically located at the site of the "custodian" (manufacturer, wholesaler, distributor, retailer or maintenance service company). An "item" may be any physical object, such as raw materials, parts and finished goods, medical drugs as well as containers used to transport other items around the world (paragraphs [0002] and [0003] of the application).

1.3 In order to optimise a supply chain, the stored events may be analysed and information may be extracted by an Information Service ("IS"). However, each custodian typically stores all events on its own, local "on-site repositories", and events relevant for the query item may be spread out on a plurality of different event servers located at their corresponding companies. Accordingly, the discovery service should be able to identify and find all events relevant to the query item.

1.4 Containers, such as pallets and boxes, are used throughout supply chains to aggregate items for transportation. The query item may therefore move "hidden" in a container through the supply chain. The query item may even be contained in a first container ("child"), which in turn may be contained in a second container ("parent"). Such higher degrees/levels of packing are referred to as packing "hierarchy", the events as "aggregation events". Accordingly, the discovery service should also be able to identify and find all events that are relevant to a container containing the query item (paragraphs [0007] to [0015]).

1.5 Discovery services ("DS") should be able to distinguish between trips of a container that are relevant for the query item and trips that are not relevant for the query item, because containers can be used many times and for different products. Falsely detected container events are called false positives. The discovery services have to be very fast, e.g. for a prescription of a medicinal drug only a few seconds are available for the query.

1.6 The present invention addresses these challenges by:

(i) analysing "event notifications", i.e. translations of the events, stored in the tracking / tracing / discovery server and performing the method on the server for running the discovery service instead of a separate system, the "event notifications" comprising an action, a parent identifier, and a child identifier (Feature (G'));

(ii) providing for a narrowing of the data space during each recursion by way of the input "time span", i.e. only events in a relevant time interval are taken into account (Features (I') and (L'));

(iii) using a combined pre-search and filter approach (Auxiliary Request, Features (M'')-(R'')).

2. **Admittance of the Main Request and Auxiliary Request into the appeal proceedings, Article 13(2) RPBA 2020**

The Appellant filed the Main Request and the Auxiliary Request after notification of a summons to oral proceedings. As they address objections raised for the first time by the board in the communication accompanying the summons, the Board is satisfied that these are "exceptional circumstances" within the meaning of Article 13(2) RPBA 2020, and admits them accordingly.

3. **Main Request**

3.1 **Closest prior art**

D8 discloses an implementation of tracking, tracing and discovery service functionality on one and the same server logic. The discovery service analyses event notifications (*abstractions*) instead of events on the repositories. D8 is therefore considered as closest prior art. D5 discloses details of an EPC system. D7 discloses in an EPC system tracing based on a dependency graph that is modeled over time.

3.2 Difference

3.2.1 D8 discloses an EPC discovery service (Fig. 5) as discussed in more detail in D5. The tracing steps are listed in sections 4 to 4.2. Tracing is performed by a so-called ONS/DS manager, comprising an Object Naming Service, an EPCISDS (Electronic Product Code - Information Service - Discovery Service) and an EPCISDS connector (Fig. 3). These components together are designed in the same logic as *one system* and are labelled here "DS". Hierarchical tracing of aggregations can be performed (Fig. 6). D8 further mentions that the EPCISDS connector transmits events and timestamps at intervals of every 10 seconds to the DS, where they are registered.

3.2.2 During oral proceedings the Appellant acknowledged that D8 disclosed in Fig. 3, Fig. 6 and sections 3.1 to 4.2 the steps of creating event notifications (*abstractions*) on the DS as claimed in Features (A')-(F'). The Appellant argued that D8 however did not disclose that each *abstraction* further comprises an action, a parent identifier, and a child identifier (Feature (G')). This was not unambiguously derivable from the passages cited above and in particular from Fig. 6. The Board agrees.

3.2.3 As to the feature that tracking, tracing and discovery service is carried out by the same application logic comprising the discovery service database (Features (B') and (C')), D8 discloses to use one system, where all data is easily available (see section 3: "*ONS and EPCISDS are designed as one system for user convenience. The system is named the ONS/DS Manager and its structure can be seen in Figure 3*").

3.2.4 It was discussed at the oral proceedings that the construction of a *complete distribution path* and the tracing method (steps (1) to (6) in section 4.2) implies most of Features (H')-(K'). The Appellant submitted that the method disclosed in D8 however did not disclose that:

(a) the event notifications comprise an action, a parent identifier, and a child identifier;

(b) the method provides for a narrowing of the data space during each recursion by way of the input time span.

3.2.5 The Board agrees with the Appellant's submissions that D8 does not disclose differences (a) and (b).

3.3 Effect

3.3.1 The Appellant argued in the statement of grounds of appeal that the effect of the above differences was an improved tracing of a tangible item in a physical supply chain. The data in the repositories did not need to be accessed in their entirety, which put load on the network and introduced the risk of a data security breach. This resulted also in that significantly less memory and computational effort was required.

3.3.2 Transferring the entire data stored in the on-site repositories to the tracing service system was also not required, which reduced load on the network, delay times and data security risks. By installing the tracing logic in the discovery server unnecessary duplication of data and logic was avoided wherein on the other side security and efficiency was increased, since all translations took place internally within the discovery service.

3.3.3 The Board agrees with the Appellant and that these effects are technical.

3.4 Problem

The problem to be solved can therefore be considered as tracing a tangible item in a physical supply chain and thereby using less computer resources - such as memory, processing time, network resources and logic - and improving security.

3.5 Obviousness

ad (a)

3.5.1 The Appellant argued that the prior art tracing method steps operated on the measurement data stored on the on-site repositories, namely the "events", whereas the claimed tracing method steps operated on "event notifications", which are stored on the discovery service. Therefore, even if the tracking functionality was implemented on the discovery service system, this would hardly improve the performance thereof, because data would still need to be fetched from the on-site repositories. D8 did not disclose to upload the event data listed in Feature (G') into the abstracts.

3.5.2 The Board however is of the opinion that D8 teaches to upload translated event data. If, according to the problem to be solved, computer power, processing time and memory space are to be reduced, the skilled person would consider to reduce the number of queries to the repositories as far as possible, because these Internet connections require time and increase the load onto the network.

3.5.3 In D8 *abstractions* of events are uploaded into the discovery service (see Fig. 6 and section 3.1, third last sentence of the last but one paragraph: "The *EPCISDS connector requests the subscription of all events to the EPCIS and transmits an abstraction of the result at intervals of every 10 seconds. The abstraction contains the URL of the EPCIS, timestamp, EPC and event type*"). The Board agrees with the Appellant that these abstractions correspond to the event notifications in the present claim and that D8 does not explicitly disclose that the abstractions contain the "action" such as "addition" and "deletion" or "parent/child information".

3.5.4 The *abstractions* however contain the information "aggregation" as shown in the inlets of Fig. 5 ("aggregation") and Fig. 6 ("General Query" -> "Event Type" = "AggregationEvent"). D8 further discloses that "[*t]he Tracing Service operates on the ONS/DS manager [...] to get the complete distribution path of the product"* (last sentence on page 556). In order to determine the *complete distribution path* a kind of dependency graph has to be construed as discussed in section II.D of D7. The *complete distribution path* however requires the information about "deletion events" and "addition events", i.e. removal of the product from a container or addition of the product to a container, as well as parent / child information.

3.5.5 This aggregation data is available at least via the URL link in the *abstractions* (Fig. 6). The Board finds that during *translation* of the event data into the *abstractions* the whole event data, i.e. comprising also action / aggregation information, has to be loaded into the virtual memory of at least one of ONS/DS, EPCICDs or EPCISDS connector. This data is therefore at least temporarily stored on the DS.

3.5.6 It is therefore obvious in view of the disclosure of D8 and the problem to be solved that any event information necessary for construing the *complete distribution path* is instantly available in the Discovery Service instead of being available only via the URLs and further querying.

3.5.7 In view of the fact that the full event data is accessed on the EPCIS and therefore saved in the virtual memory of the DS, it would be wasteful - and contrary to the problem to be solved - not to save these data into a permanent memory of the DS, since this data is needed later for construing the *complete distribution path.*

3.5.8 In order to reduce the network load and processing time while providing the *complete distribution path* it would therefore be necessary to keep the required event information available in the abstractions on the DS instead of having to repeatedly fetch event data from the on-site repositories.

3.5.9 Therefore, Feature (a) is obvious in the light of the disclosure and teaching of D8.

ad (b)

3.5.10 If computer power, processing time and memory space are to be reduced, the skilled person would first consider to search only events which are relevant with respect to object number, action and time span. The skilled person would therefore consider only relevant object identifiers and timestamps within a given relevant input time span (time interval) and in relevant locations. As known from everyday life, nobody would search a front door key at places outside their house and at places where they have been the day(s) before, if the key was lost at the same day after opening the front door of the house. The skilled person would accordingly use only time spans and parent/child identifiers relevant to the events, e.g. consider only events of a relevant day and not all the other 364 days of the year.

3.5.11 D8 teaches to consider timestamps but does not teach to query for specific time intervals (see section 3.5.3 above). D7 however teaches that a *"query can be controlled by several parameters, e.g. time intervals, EPCs, locations etc."* (last but one sentence of section II.D). D7 furthermore teaches to iteratively construe a dependency graph (see section II.D). The *complete distribution path* of the product mentioned in D8 is such a *dependency graph*. Independent of the teaching of D7, it is common sense that only those events are considered which are within a relevant time span / interval.

3.5.12 It is, for example, evident that it would make no sense to consider events related to a container for a time span where a product could not be inside the container. If a product is added to a container at a time x for the first time, one does not have to consider container events earlier than x. If a product was definitely removed from a container at a time y, it also does not make any sense to consider any events related to this container later than y. It is therefore an obvious measure to adapt the relevant time span for each container and to evaluate the relevant time span in each recursive step.

3.5.13 The container time intervals ("input time span") therefore become narrower during iteration. For example, when tracing a product over its complete distribution path, it would have spent a shorter time span on a ship than that it spent inside a shipping container, which again is a shorter time span than that inside the product packaging. It follows that a method taking into account only relevant time spans provides for a narrowing of the data space during each recursion by way of the input time span.

3.5.14 Therefore, Feature (b) is obvious in the light of the disclosures and teachings of D7 and D8. Consequently, the subject-matter of claim 1 of the Main Request does not involve an inventive step within the meaning of Article 56 EPC.

4. **Auxiliary Request**

4.1 Closest prior art

Although D7 discloses several added features corresponding to an iterative method of construing a dependency graph, D8 is still considered to be the closest prior art.

4.2 Difference

Features (M'')-(R'') are not disclosed in D8.

4.3 Effect

4.3.1 The claimed method first applies a (pre-)search method and then subsequently applies a filter method that filters out any remaining false positives. Focusing on steps (I')-(L') on specific time spans has the consequence that event notifications may be extracted which correspond to containers having contained the traced product at a certain moment in a relevant time span, but in reality did not contain the product, although detected "positively" in the algorithm of (I')-(L').

4.3.2 The method defined in the Main Request has the drawback that too many false positives may be generated (statement of the grounds of appeal, section C, first paragraph). The problem of false positives for EPC discovery services is well-known (application, paragraph [0072]; D5, section C.3.2, last but two paragraph, and section C.4.1, last but one bullet point).

4.3.3 These false positives are filtered out in a second filter method as defined in steps (M'')-(R'') of the Auxiliary Request.

4.4 Problem

The problem can be formulated as filtering out false positives being a result of the filter method defined in the Main Request. This problem can be assessed independently from the problem formulated in section 3.4 above, but is a direct consequence of it.

4.5 Obviousness

4.5.1 In order to eliminate irrelevant container events, the additional filter algorithm of the Auxiliary Request corresponds to a stack algorithm which construes in chronological order a dependency graph of the transport chain by maintaining only relevant events. By following the *distribution path* for each event notification from the subset filtered out in the filter method (steps (H')-(L')) it is verified whether *the current event notification is an object event notification and its object identifier is stored somewhere in the stack* (Feature (R'')).

4.5.2 Features (M'')-(R'') correspond to a stack keeping track of the hierarchy of containers containing an item of interest at the time of the current event in the list of event notifications. However, since the hierarchy of containers containing the element of interest changes in a last-in-first-out manner (the last closed container is the first opened one) - which per definition is the manner in which a stack operates - it is straightforward to use a stack in order to keep track of this hierarchy while processing the sequence of events.

4.5.3 This realisation is common sense, namely, that the outermost container - which is the last one into which an item has been enclosed - is the first one to be opened. Knowing that the hierarchy of containers changes in a last-closed-first-opened manner and needing thus to keep track of the hierarchy that changes in a last-in-first-out manner, the skilled person would straightforwardly resort to a stack data structure since this is the data structure to be used for information that changes in a last-in-first-out manner (see also D7, section II.C and the algorithm disclosed in Fig. 4).

4.5.4 D7 teaches to perform a single in-order pass over the set of chronologically ordered events ("*The events are propagated through all their descendants in the graph*"; "*we maintain a dependency graph holding at each point in time the current relations between EPCs while we* *replay the event log in chronological order*", page 288, left column, second and third paragraph, Fig. 4). In order to find the items to which a certain event is relevant, the hierarchy of containers at the current time in the stream of events needs to be maintained in order to propagate an event related to an item to all its current descendants, i.e. to all items contained by the item. In D7 the tree data structure that needs to remember the hierarchy of containers at a certain time only needs to remember the containers enclosing the item of interest, wherein the tree structure becomes in fact a stack as claimed. Also, only the associations between the queried item of interest and its relevant events need to be remembered.

4.5.5 The same section of D7 also teaches to use a two-step approach. In a pre-search step relevant events are mapped into a MXML file (Fig. 3) by querying events from an EPCIS repository (the "Mining XML" MXML format is a generic XML based format designed to store log files). In a second step a workbench filters out the relevant elements from the pre-selected events in the MXML file and construes the dependency graph. It is therefore a normal option to query in several stages.

4.5.6 The Appellant argued that claim 1 of the Auxiliary Request comprised a plurality of complex features not explicitly disclosed in the prior art and that there were no pointers in the prior art towards the solution as defined in the claims. As was reasoned in the decision of the German Federal Court dated 7 September 2010, file number X ZR 173/07 ("Walzgerüst II"), Reasons 36, the suggestion required for an obviousness of the solution according to the invention could not be replaced by the factual logic of the proposed solution of the invention.

4.5.7 Steps (N'')-(R'') do not define independent complex technical features, but a single last-in-first-out stack algorithm following the packaging hierarchy according to the logic of D7 and common sense. In decision T 914/02, Reasons 2.3.4, it was held that "*it is doubtful as a matter of principle whether complexity can be used to disqualify an activity as a mental activity*". Accordingly, in the present case it could be argued that it is doubtful as a matter of principle whether a complex wording of a straightforward algorithm can be used to disqualify the algorithm as being obvious.

4.5.8 It is therefore not a hindsight analysis to conclude that starting from the disclosure of D8 - in order to remove the false positives - the skilled person would consider an iterative method for construing the dependency graph as suggested in D8 (*complete distribution path*) in combination with D7 (*dependency graph*) and straightforwardly use a stack to keep track of the hierarchy of containers.

4.6 **To summarise** the subject-matter of claim 1 of the Auxiliary Request is not inventive within the meaning of Article 56 EPC in view of the teaching of D8 in combination with the teaching of D7 and the common general knowledge.

Order

For these reasons it is decided that:

The appeal is dismissed.

The Registrar: The Chairman:

Beschwerdekammern des Europäischen Patentamts
Boards of Appeal of the European Patent Office
Chambres de recours de l'Office européen des brevets

S. Sánchez Chiquero G. Eliasson

Decision electronically authenticated