Data State Tracking: labelling good quality data to improve warehouse operations

PHILIP WOODALL, University of Cambridge
VAGGELIS GIANNIKAS, University of Cambridge
WENRONG LU, University of Cambridge
DUNCAN MCFARLANE, University of Cambridge

Warehouse Management Systems (WMSs) record where products are stored in a physical warehouse; however, when pickers misplace products the WMS becomes misaligned with the real situation. Due to the fact that the WMS is not aware of these misalignments, it can lead pickers to empty shelves or shelves with the wrong and unexpected products. This paper proposes an extension to a WMS that tracks the state of the quality of the data and modifies picking reports based on this state to help pickers either avoid or discover these misalignments, depending on how it is configured. Using an experimental simulation to evaluate the approach, this paper shows how the solution can outperform a standalone WMS by approximately 1% when trying to avoid misalignments, and by approximately 32% when attempting to discover the misalignments.

1. INTRODUCTION

Existing information systems assume that the data they provide is 100% correct in every dimension of data quality; they give the user no indication about potential errors in the data when reports are requested from the system. For example, in the warehousing domain, warehouse management systems (WMS) are used to record the locations of physical products in the warehouse, and they always return a single and exact location at which the product resides. The problem is that this location is not always correct because items can be accidentally placed in the wrong locations [Rekik et al. 2008] [Lee and Özer 2007]. Therefore, the WMS does not, in fact, know exactly where the products are despite it giving the appearance that it knows with 100% certainty. The consequence of the WMS performing in this way is that it can cause operational disruptions, such as inadvertently leading staff to 1) warehouse locations that contain too few products (or are empty), or to 2) locations that contain the wrong products (the latter can sometimes result in shipping the wrong products to the customer). These WMS-related disruptions are caused by data *misalignments*, which are defined as accuracy problems where the data in the information system (WMS) does not align with the state of the physical world [Wand and Wang 1996]. Note that it is not that the data is wrong, nor that reality is wrong (!), but rather that the two are not in alignment (note that either can be corrected to alleviate the misalignment).

To address WMS-related data misalignment problems, the Data State Tracking (DST) capability is proposed, which builds on the concept of probabilistic databases [Suciu et al. 2011]. This capability can be attached to a WMS in order to:

1. Record and track definite and potential problems with the data,
2. Support the automatic or manual correction of data misalignments,
3. Produce reports that influence operations in a way that can help to avoid or discover data misalignments.

In this paper we show how requirements one and three of the DST capability have been implemented to either avoid or discover cases of the WMS misalignments, which cause the above warehouse disruptions. Discovery is useful to help rectify the problems when the warehouse has available resources (e.g. spare time), and

avoidance is useful when the warehouse is busy. A software simulation was developed to compare the performance of the DST capability against a normal WMS, and the results show that DST is able to outperform a normal WMS when both avoiding and discovering these disruptions.

This paper proceeds by describing alternative and existing methods for addressing misalignments between a WMS and the items in a warehouse. Section 3 illustrates how the misalignments can arise. Section 4 describes the DST concept in detail including how a WMS can be extended to include this capability, and sections 5 and 6 presents the experimental simulation (configuration and results) used to evaluate a WMS with the DST capability attached. Finally, section 7 presents the conclusions of the paper.

2. ADDRESSING INVENTORY ACCURACY IN WMS SYSTEMS

A key requirement of a WMS is to reduce misalignments (also referred to as discrepancies in the literature) and provide accurate information on inventory and storage locations [Davarzani and Norrman 2015]. Inventory checking and electronic tagging of items are two existing ways of reducing misalignments. For electronic tagging, Radio Frequency Identification (RFID) tags are commonly used to keep track of exactly where the items are [Zhou et al. 2015]. However, RFID is an expensive option that cannot always be afforded to be used [Vijayaraman and Osyk 2006]. Moreover, RFID for inventory checking can be unreliable [Zhao and Ng 2012]. Inventory checking is another method and the key is determining how frequently to count [Rekik 2011] [Iglehart and Morey 1972]; this is because inventory checking an entire warehouse is an expensive operation especially in terms of the time and resources required; methods have therefore been developed to reduce this [Rossetti et al. 2001].

From an information quality viewpoint, traditional data quality assessment improvement initiatives (see [Batini et al. 2009; Woodall et al. 2013]) do not apply well to the picking operations in a warehouse environment because of the rate at which the data in the system and products in the warehouse change and can become misaligned. Inventory checking is essentially the data quality assessment and improvement exercise in this case. Finally, probabilistic databases are able to model the case where a database instance could be in one of several states (with each state having a probability indicating the likelihood that the state is correct) [Suciu et al. 2011]. Therefore, the tuples in a WMS could be modelled probabilistically to indicate the chances that they may not represent exactly where the products reside. This type of mechanism clearly satisfies the first requirement of the DST capability, and hence the DST solution builds on this work. The only difference with the DST approach for the case in this paper is that it uses discrete states rather than probabilities.

3. CAUSES OF DATA MISALIGNMENTS IN WAREHOUSES

One of the causes of the two disruptions of too few products or incorrect products (mentioned above) is “incorrect product replacement” [Lee and Özer 2007] [Rekik et al. 2008]. This problem occurs when there is an empty location (shelf) nearby where the picker accidentally replaces any remaining products that need placing back on the shelf. Figures 1 and 2 illustrate this problem.

Figure 1 shows a static snapshot of the state of the WMS system (on the left) and the physical warehouse (on the right). The WMS contains data records that indicate that there are 20 kettles stored in location A, 15 toasters stored in location B, and 5 kettles stored in location C. Locations D and E are recorded as being empty shelves. Equally, the warehouse physically contains 20 kettles in location A etc.

In Figure 2, a new order is received for 17 kettles. These are picked from location A but the remaining 3 are misplaced into location D. Hence, after the WMS is updated there is now a misalignment between the data and the warehouse.

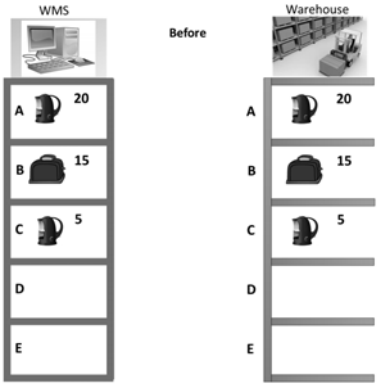

Fig. 1. A static snapshot of the state of the WMS and warehouse.

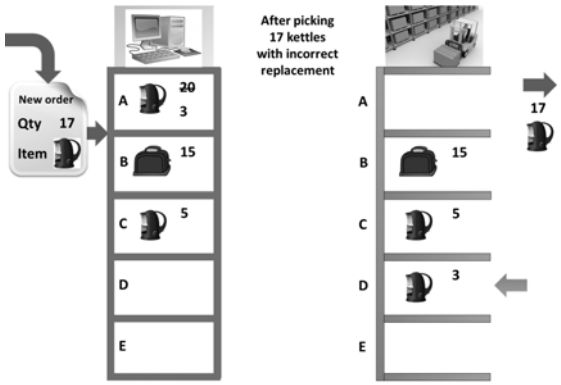

Fig. 2. An illustration of the incorrect item replacement scenario.

4. IMPROVING WMS PERFORMANCE VIA DATA STATE TRACKING

In this section we describe how to improve a WMS by attaching to it a Data State Tracking capability, which is defined as an extension to a relational database which is able to:

1. Record and track definite and potential problems with the data,
2. Support the automatic or manual correction of data misalignments,
3. Produce reports that influence operations in a way that can help to avoid or discover data misalignments.

As can be seen in the architecture of the DST attached to a WMS (shown in Figure 3), the Data State Tracker, Data State Model and the data state itself are used to achieve the first requirement. The Correction Engine is used to satisfy the second requirement, and the Reporting Engine satisfies the third requirement.

The "data state" records metadata about the potential and actual problems with the tuple or particular data value of interest in the database and any other metadata that may be required to help correct the WMS data. In a WMS, tuples contain the attributes product type, quantity and location (and sometimes product condition [Rossetti et al. 2001]). Hence, the data state is used to capture when these values are definitely or potentially misaligned with the state of reality. In this case, the data state is physically stored as an extra field in the database. The data states will vary

depending on the problem being addressed; as an example, for the WMS case, a general set of data states are:

- State 1: data is reliable (e.g. the data has been confirmed to be correct after an inventory check)
- State 2: data is potentially unreliable (e.g. products could have potentially been misplaced in the warehouse, so the product type, quantity and location may not be accurate).
- State 3: data is definitely unreliable (e.g. a picker has reported a misalignment in the warehouse after trying to pick the products)

Clearly, these could also be represented by probabilities (as is the case in probabilistic databases) in cases where it is possible to determine continuous values for data reliability. It can be seen in the following evaluation section that for the experimental situation considered, it is sufficient to use discrete states rather than continuous values.

The data state model (see the right hand side of Figure 3) defines these states and formally describes when to switch between them. The Data State Model includes, for each transition arrow, an event (e.g. an event occurring during physical operations, or an event in the information system) which indicates when to make the transition, and a condition defining whether to make the transition or not.

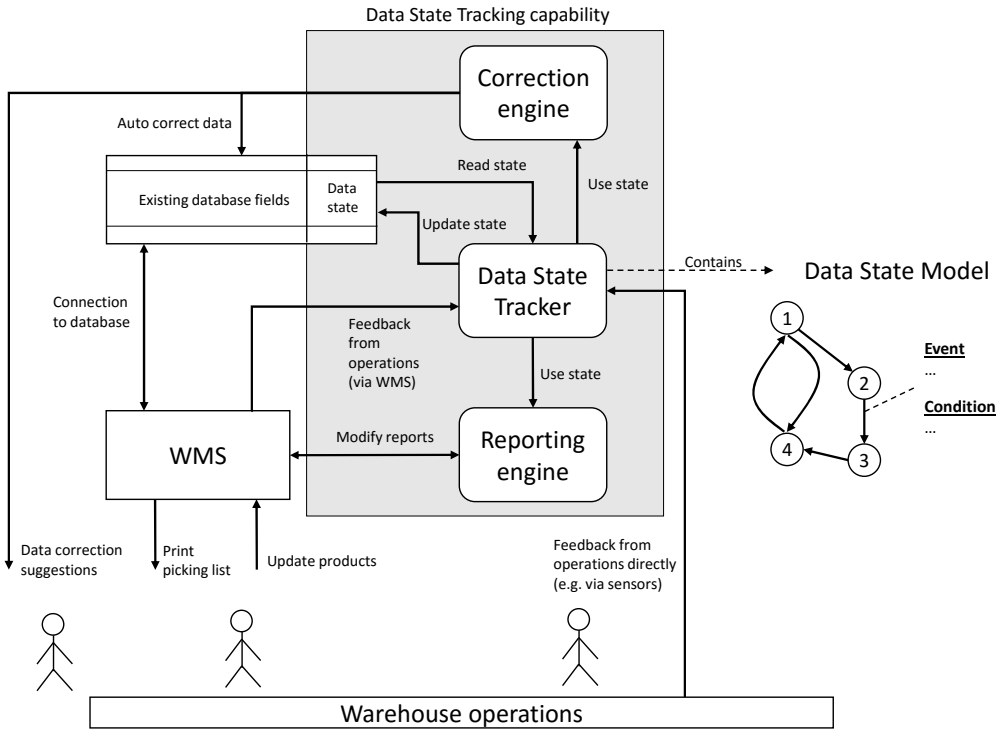

Fig. 3 The architecture of the DST coupled to a WMS

The Data State Tracker component uses feedback from operations either directly (e.g. via sensing, videos etc.) or via the WMS to update the data states by applying the

Data State Model. This component has the responsibility to read and update the data states attached to the database.

The Reporting Engine has the capability to modify the reports generated by the WMS, by considering the data states, and feed those back to the WMS to present as its final output. Normally, a WMS generates a picking list by considering all the possible locations a product could be picked from, and selecting from those. With the DST capability, the Reporting Engine can modify this result to help avoid or discover misalignments. For example, in order to avoid disruptions the picking list can be generated by prioritising the picking of products from locations having a state that indicates reliable data. Similarly, in order to discover misalignments, the reverse is used: states that indicate unreliable data can be prioritised. Both of these are possible because warehouses usually have various degrees of freedom regarding which locations products can be picked from (i.e. there are multiple locations containing the same product type). The DST capability relies on having these degrees of freedom to operate and it essentially suggests the best options to take within these degrees of freedom (which would otherwise be considered equal).

The aim of the Correction Engine is to make suggestions to users about how data can be corrected, or to actually make the corrections directly and automatically, if possible. As an example, the Correction Engine could keep track of potential locations where the products may have been misplaced. Then, if a picker observes a problem directly (e.g. encounters an empty location that is not supposed to be empty), the Data State Tracker could use a state to indicate that a definite misalignment has been found at the location (as opposed to a potential misalignment) to prompt the Correction Engine to investigate this. The Correction Engine could then attempt to locate the missing products based on the potential locations it has been recording. In the deterministic case, the data could be corrected automatically, and when it is not clear exactly where the missing products are, suggestions can be made to the user to support an inventory checking activity. Furthermore, the Correction Engine could make recommendations for where to find missing products based on an analysis (e.g. data mining) of historical data on where items have been misplaced.

5. WAREHOUSE SIMULTATION STUDY

In this paper we have implemented and evaluated part of the DST capability for a basic case involving the problems caused by pickers misplacing products in a warehouse. Only the Data State Tracker and Reporting Engine have been implemented and a simple data state model consisting of two states is used. This is the most basic case because the states represent reliable data and potentially unreliable data. As we have not considered the correction of data in this evaluation, we do not use any states that indicate definitely unreliable data.

The aim of the evaluation was to measure, for a given number of warehouse orders, whether the aforementioned two warehouse disruptions could be avoided or discovered more than a standalone WMS when using the WMS with DST capability configured to avoid or discover misalignments respectively.

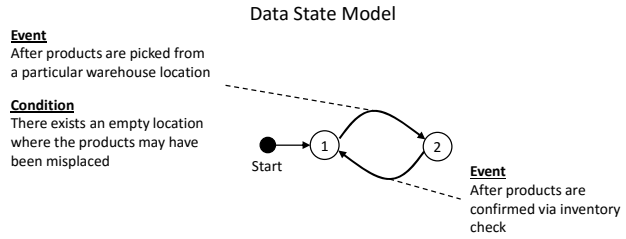

Fig. 4. The implemented data state model

To conduct this evaluation, an experiment was carried out using a software simulation of picking operations, which included a simplistic WMS system and a warehouse system. The DST capability was also developed in software (using Java and MySQL). The experiments were conducted with and without the DST capability applied, hereon referred to as the "normal" and "solution" cases respectively.

The state model used is shown in figure 4 where initially all tuples start in state 1 (after an inventory check). Tuples move to state 2 after the picking products from that tuple if it is possible to misplace them (i.e. there exists an empty location in the warehouse where they can be misplaced). In the future, this basic case could be extended to consider only nearby empty locations as potential places where products could be misplaced and the labelling of them with probabilities. To avoid disruptions the picking list was modified to give priority to locations with state 1 over state 2. In the tests to discover misalignments, state 2 was given priority. Table I details the configuration of the experiment, which consisted of 10 trials each with 200 orders. For each trial the only variable was the insertion of errors: these were varied randomly by attempting to misplace products into empty locations at random orders (see table II).

Table I. Experiment configuration

Warehouse locations	100 (Each is unique and only one product type is allowed in one location)
Number of different product types	5
Initial state of the warehouse and WMS	No misalignments, random insertion of product type (1-5), random quantity (1-100), random placement into locations. Reset for each trial.
Orders (picking tasks) per trial	200 per trial. The order contains one random product type (1-5) and quantity (1-20). There are only outbound orders (hence there are no storing operations)
Number of trials	10 (the WMS and warehouse systems are reset after each trial)
Number of empty locations	20 (randomly distributed in the warehouse)
Insertion of errors (misalignments)	10 (attempted at random picking tasks). Some may not be successful if there is nothing to pick for that task or there are no empty locations available. The first empty location given by the warehouse system is chosen to misplace the products. The insertion of errors differed randomly for each trial.
Satisfying orders	Pickers only pick from locations that have the full quantity available to satisfy the order. The WMS only selects single locations in this manner also.
Picking policy	The actual location from which items are picked is chosen as the first location in the list generated by the WMS in solution mode. Note that this location will always be the same in trials for both the normal and solution cases.
Misalignment correction	Even if a disruption is found, it is not corrected, nor recorded on a "black list" for future picking to avoid.

In the simulation, the missing products disruption was measured by recording the probability of the picker encountering a location with no products or too few products. Similarly, the incorrect products disruption was measured by recording the probability of the picker encountering a different product type at a location. These were aggregated into a single measurement to represent the probability of facing a disruption.

6. PERFORMANCE EVALUATION RESULTS

This section presents the results of the experiments (using the configuration described in the previous section). Firstly when trying to avoid disruptions and then when setting the reporting priority to discover disruptions.

6.1 USING THE "AVOID DISRUPTIONS" STRATEGY

Note that each trial from 1 to 10 differed by the random insertion of product misplacement errors. The results of trials for the normal and solution cases and the attempted and actual misplacements are shown in Table II, with the order numbers in bold for the actual misplacements. As an example of why it was not always possible to misplace the products, in trial 7 the wrong product type was encountered during order number 82, and so no items were picked.

For each experimental trial figure 5 shows that the solution always performs better, and the mean improvement over all trials is approximately 1% (0.97%: the difference between the means in table II). The maximum improvement of approximately 1.66% can be seen in trial 4, and the minimum improvement of approximately 0.27% can be seen in trial 5.

Further experiments found that the solution performs worse when the errors are skewed towards later orders, rather than evenly distributed throughout the orders. Experimental trials 11 to 14 deliberately skewed the distribution of errors to be either in the first 100 orders or the second 100 orders. Table III shows which orders the misplacements were attempted and the results after executing the normal and solution cases; the means are as before: the mean probability of encountering a disruption over 200 orders).

Table II. Results for the avoid disruptions strategy

Trial	Order numbers which misplacements were attempted (actual misplacements are shown in bold)	Actual number of errors inserted	Mean probability (normal)	Mean probability (solution)	Difference
1	**6**, **10**, **103**, **118**, 123, **142**, 156, **165**, 187, 199	6	0.093111	0.085248	0.0079
2	**52,** 56**, 60, 73, 75,** 77, **108, 136, 186, 194**	8	0.076651	0.073851	0.0028
3	**19, 31, 42, 78, 90,** 115**, 153, 154,** 160, 188	7	0.102756	0.08987	0.0129
4	**18, 31,** 36**, 38, 50, 74, 91, 102,** 144, 180	7	0.078549	0.06196	0.0166
5	**15, 17, 46, 73, 126, 149, 161,** 164, 171, 180	7	0.054996	0.052284	0.0027
6	**6, 26, 29, 42,** 143**, 145, 159,** 168**, 179, 186**	8	0.099245	0.088285	0.0110
7	**22, 40, 57, 78,** 82**, 102, 103,** 107**, 136,** 172	7	0.101829	0.092445	0.0094
8	**1, 2, 25, 51, 112,** 122**, 129, 150,** 168**, 177**	8	0.072032	0.059689	0.0123
9	**22, 23, 66, 69, 78,** 93**, 99, 103,** 111**,** 172	7	0.086211	0.074404	0.0118
10	**23,** 28**, 84, 114, 134, 152, 155,** 160, 169**,** 191	6	0.05723	0.053661	0.0036
Mean	-	-	0.082261	0.07317	0.0097

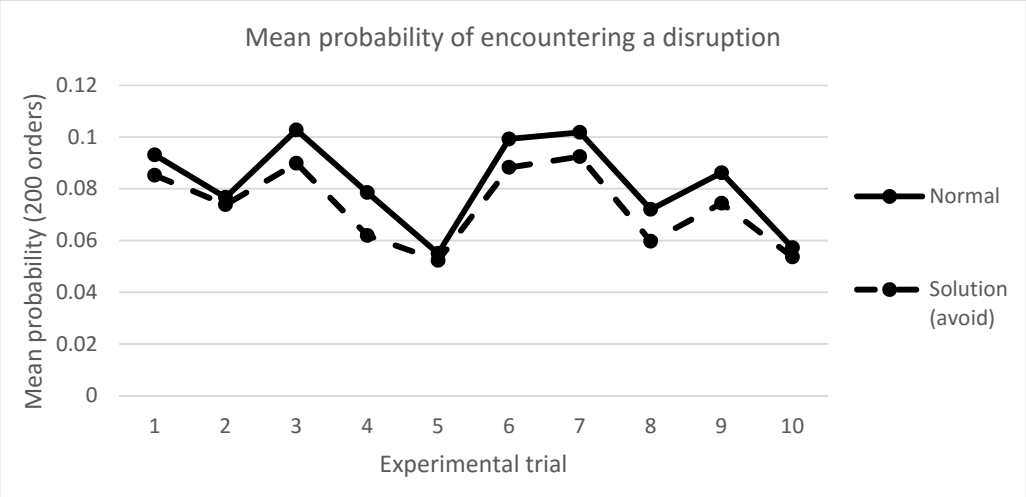

Fig. 5. The mean probability of encountering a disruption for the normal and solution cases when trying to avoid disruptions

Table III. Mean avoidance results for the insertion of errors in the first 100 or second 100 orders

Trial	Order numbers which misplacements were attempted (actual misplacements are shown in bold)	Actual number of errors inserted	Mean (normal)	Mean (solution)	Difference
11	**101**, **102**, 122, **126**, **128, 140, 148**, 156, 175, 197	6	0.049244	0.046527	0.0027
12	**107**, **117**, 126, **127**, 129, **133**, **149**, 152, 154, 187	5	0.054727	0.054727	0
13	**6, 25**, 28, 36, **37**, **46, 59, 69, 70, 91**	8	0.07069	0.051019	0.0197
14	**31, 41, 43, 54, 57, 58**, 75, **79**, **92,** 95	8	0.133321	0.122584	0.0107

Trials 11 and 12 are for misplacements that occurred in later orders and these show how the solution provides the least benefit over the normal case. In fact, for trial 12, there is no benefit and the solution operates with exactly the same performance as the normal case. The opposite is true when the misplacements occurred in earlier orders: the solution performs almost up to a 2% improvement (see trial 13) compared to the normal case.

In trial 12, the first inventory error was inserted at order number 107 (product type 4 was misplaced), and the first disruption was faced at order 111, which was the first time since the error that an order requested product type 4). There were no locations having state 1 that contained product type 4, and so the solution could do no better than the normal case. This situation continued for the other product types resulting in no overall performance gain for the solution. The reason the solution operates with exactly the same performance as the normal case in trial 12 is because the solution becomes the normal case when there are no locations which are state 1 (i.e. assured to be free from error) to pick an order from. This is why the solution performs worse when errors occur in later orders: there are cases where there are no state 1 locations to pick from because they have turned into state 2 locations. With more orders this becomes more likely.

6.2 USING THE "DISCOVERING DISRUPTIONS" STRATEGY

The solution was altered to deliberately discover disruptions by setting the Reporting Engine to return state 2 locations before state 1 locations in a picking list. The effects after running the same experimental configuration to generate the results in figure 5 are shown in figure 6. The solution was able to discover disruptions with a mean of approximately 32% improvement over the normal case (the difference between the two means in Table IV). The results show a maximum improvement over the normal case of approximately 48.4%, in trial 2, and minimum improvement of approximately 18.3%, in trial 10.

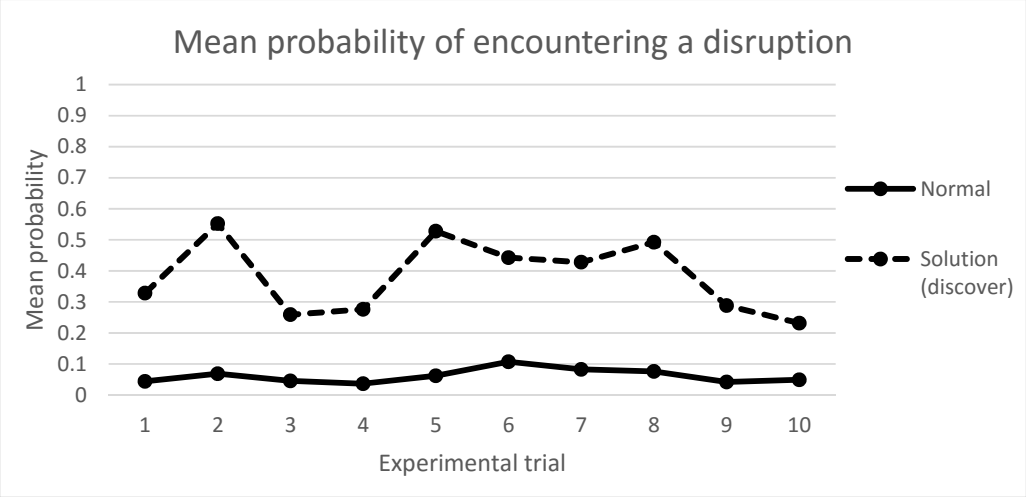

Fig. 6. The mean probability of encountering a disruption for the normal and solution cases when trying to discover disruptions

Table IV. Results for the discovering disruptions strategy

Trial	Order numbers which misplacements were attempted (actual misplacements are shown in bold)	Actual number of errors inserted	Mean probability (normal)	Mean probability (solution)	Difference
1	**6**, **10**, **103**, 118, 123, **142**, 156, **165**, 187, 199.	5	0.044337	0.328652	0.284
2	**52,** 56**, 60,** 73**, 75,** 77, **108, 136,** 186, 194	5	0.068789	0.5525	0.484
3	**19, 31,** 42**, 78,** 90**,** 115**, 153, 154, 160**, **188**	7	0.045868	0.259052	0.213
4	**18, 31, 36, 38,** 50**, 74, 91,** 102**,** 144, 180	6	0.036685	0.276339	0.240
5	**15, 17,** 46**, 73,** 126**, 149,** 161**,** 164, 171, 180	4	0.062111	0.527917	0.466
6	**6, 26, 29, 42, 143,** 145**, 159,** 168**, 179, 186**	8	0.107464	0.442959	0.335
7	**22, 40,** 57**, 78,** 82**, 102, 103, 107,** 136**, 172**	7	0.082681	0.428143	0.345
8	**1,** 2**, 25, 51, 112,** 122**, 129,** 150**,** 168**,** 177	5	0.076349	0.4925	0.416
9	**22, 23, 66, 69,** 78, 93, 99**, 103,** 111**, 172**	6	0.042194	0.288434	0.246
10	**23, 28,** 84**, 114, 134, 152, 155,** 160, **169, 191**	8	0.048973	0.231902	0.183
Mean	-	-	0.061545	0.38284	0.321

Again, the additional trials 11 to 14 found that that the solution performs worse when the errors are skewed towards later orders, rather than evenly distributed throughout the orders (see Table V). Although, in the worst case, the solution outperformed the normal case by 11.6% (trial 12) and by 45.1% (trial 14) in the best case—overall much greater margins than in the avoid case.

Table V. Mean discovery results for the insertion of errors in the first 100 or second 100 orders (bold indicates actual insertion of the error)

Trial	Order numbers which misplacements were attempted (actual misplacements are shown in bold)	Actual number of errors inserted	Mean (normal)	Mean (solution)	Difference
11	**101**, **102**, 122, **126**, 128, 140, 148, 156, 175, 197	3	0.044308	0.220812	0.177
12	107, **117, 126**, 127, **129**, **133, 149**, 152, 154, 187	5	0.021741	0.137719	0.116
13	**6, 25, 28, 36, 37, 46**, 59, 69, 70, 91	6	0.052521	0.356137	0.304
14	**31, 41, 43, 54**, 57, **58**, 75, 79, **92**, 95	6	0.074438	0.525	0.451

7. CONCLUSIONS

This paper describes a capability that can be added to a WMS in order to improve its performance in terms of keeping it aligned with the actual state of the warehouse and better enabling it to choose accurate data to present to the user. So far, part of this capability has been implemented and evaluated that can 1) record and track potential problems with warehousing data, and 2) modify picking list (reports) that influence operations in a way that can help to avoid or discover data misalignments. The solution performed marginally better than a normal standalone WMS when avoiding misalignments (by approx. 1%) and much better when attempting to discover misalignments (by approx. 32%). The solution performance depends mostly on the number of degrees of freedom in the WMS for each product i.e. the number of options (number of different locations from which the product can be picked from) that the WMS has available to satisfy an order for the product. The performance also improves the closer the errors occur to a prior inventory check. Note that the solution never performs worse than the normal case as it simply reverts to the normal case if there are no data records with a state 1 (accurate data) to choose to pick items from.

As it can be observed, there is a significant difference between the benefits offered by the solution in the "avoiding disruptions" strategy compared to the "discovering disruptions" strategy. The reason relates to the small number of misalignments occurred compared to the number of shelves an item can be picked from. For the first strategy, even without the solution in place, there is a high chance a disruption will be avoided since there are many shelves an item can be picked from without causing a disruption. Here, the solution does not contribute much as the normal case already performs well. For the second strategy, discovering disruptions "by luck" in the normal case will be hard since there are not many misalignments in the warehouse (in absolute numbers). However, the solution provides some indication of what locations are "risky", which ultimately offers a greater benefit.

Although we have applied the solution to a WMS, we assert that it could also be used for other types of IT systems and physical operations. An example case could be in engineering asset management where assets are maintained and inspected (problems such as missing assets occur), and Master Data Management where, for example, customer data records need to be aligned with one another to produce a single point of truth. Furthermore, the solution could be extended to incorporate probabilistic values rather than discrete states to indicate the probability of products being misplaced.

ACKNOWLEDGMENTS

The authors would like to thank YH Global for supporting the project within which this research was developed.

REFERENCES

C. Batini, C. Cappiello, C. Francalanci, and A. Maurino. 2009. Methodologies for Data Quality Assessment and Improvement. *ACM Comput. Surv.* 41, 3 (2009), 1–52.

H. Davarzani and A. Norrman. 2015. Toward a relevant agenda for warehousing research: literature review and practitioners' input. *Logist. Res.* 8, 1 (January 2015), 1–18. DOI:http://dx.doi.org/10.1007/s12159-014-0120-1

D. Iglehart and R. Morey. 1972. Inventory Systems with Imperfect Asset Information. *Manag. Sci.* 18, 8 (April 1972), B-388-B-394. DOI:http://dx.doi.org/10.1287/mnsc.18.8.B388

H. Lee and O. Özer. 2007. Unlocking the Value of RFID. *Prod. Oper. Manag.* 16, 1 (January 2007), 40–64. DOI:http://dx.doi.org/10.1111/j.1937-5956.2007.tb00165.x

Y. Rekik. 2011. Inventory inaccuracies in the wholesale supply chain. *Int. J. Prod. Econ.* 133, 1 (September 2011), 172–181. DOI:http://dx.doi.org/10.1016/j.ijpe.2010.02.012

Y. Rekik, E. Sahin, and Y. Dallery. 2008. Analysis of the impact of the RFID technology on reducing product misplacement errors at retail stores. *Int. J. Prod. Econ.* 112, 1 (March 2008), 264–278. DOI:http://dx.doi.org/10.1016/j.ijpe.2006.08.024

M. Rossetti, T. Collins, and R. Kurgund. 2001. Inventory Cycle Counting–A Review. In *The proceedings of the 2001 Industrial Engineering Research Conference.* 457–463.

D. Suciu, D. Olteanu, R. Christopher, and C. Koch. 2011. *Probabilistic Databases* 1st ed., Morgan & Claypool Publishers.

B.S. Vijayaraman and B. Osyk. 2006. An empirical study of RFID implementation in the warehousing industry. *Int. J. Logist. Manag.* 17, 1 (January 2006), 6–20. DOI:http://dx.doi.org/10.1108/09574090610663400

Y. Wand and R. Wang. 1996. Anchoring Data Quality Dimensions in Ontological Foundations. *Commun. ACM* 39, 11 (1996), 86–95. DOI:http://dx.doi.org/10.1145/240455.240479

P. Woodall, A. Borek, and A. Parlikad. 2013. Data quality assessment: The Hybrid Approach. *Inf. Manage.* 50, 7 (2013), 369–382. DOI:http://dx.doi.org/10.1016/j.im.2013.05.009

P. Woodall and A. Wainman. 2015. Data Quality in Analytics: Key Problems Arising from the Repurposing of Manufacturing Data. In *International Conference on Information Quality (ICIQ).* Cambridge, MA., 174–184.

Z. Zhao and W. Ng. 2012. A model-based approach for RFID data stream cleansing. In ACM Press, 862. DOI:http://dx.doi.org/10.1145/2396761.2396871

J. Zhou, H. Zhang, and H. Zhou. 2015. Localization of pallets in warehouses using passive RFID system. *J. Cent. South Univ.* 22, 8 (August 2015), 3017–3025. DOI:http://dx.doi.org/10.1007/s11771-015-2838-6