Mirror, mirror on the wall, can I count on you at all? Exploring data inaccuracy in business processes

Pnina Soffer

University of Haifa, Carmel Mountain 31905, Haifa, Israel
spnina@is.haifa.ac.il

Abstract. Information systems in general and process aware information systems in particular support the execution of business processes. This support is based on the assumption that the information system truly reflects the state of the domain where the process takes place. Based on this assumption, humans do not need to directly "sense" the state of affairs. Rather, decisions are made based on the state as reflected in the information system. This paper explores the situation where this assumption does not hold, namely, the situation of data inaccuracy. In particular, it formalizes data inaccuracy and addresses three questions: (a) how is data inaccuracy manifested in a process? (b) what are the expected results of data inaccuracy? (c) how can robustness to data inaccuracy be increased? The understanding gained with respect to these questions should form a basis for designing processes to be more robust, avoiding problems due to data inaccuracy.

Key words: Data inaccuracy, Process design, Generic Process Model

1 Introduction

Business process management has attracted the attention of both business and academia in the past two decades. Typically, the focus of research has been on process aware information systems and workflow management systems. However, business processes are usually performed by humans who use resources. Information systems provide different levels of support to business processes. The basic support is a simple reflection of the activities that are performed and their effect on the state of the organization and its environment. At the high end of support, process aware information systems enact, coordinate, and manage these activities, while reflecting their effect. At all levels, the basic assumption underlying business process support, is that the information system truly reflects the state of affairs. Based on this assumption, humans do not need to physically sense the current state for deciding what activity to perform at a given moment and how to perform it. However, it is well known that the information which exists in an information system is not always completely reliable [1].

The question is what would happen to the business process if and when the information its execution relies upon does not truly reflect reality; will it be able to complete? What would be the results? Clearly, there is no one answer to this question, as the answer is highly situation dependent. To illustrate, consider the following two

scenarios. The first one addresses a process of quotation preparation, which includes the analysis of work and materials required for fulfilling the customer's needs, estimating their cost, and determining the price to be charged based on business considerations. The goal of the process is reaching a state where the quotation is ready and sent to the customer. Now assume the material cost has been falsely recorded in the information system and does not reflect the actual price. This would not stop the process from achieving its goal, and the result of this inaccuracy may even remain unknown. It might, however, have an effect on the business value achieved by the process (under-pricing will harm profitability, while over-pricing might lead to rejection by the customer).

The second scenario addresses a process where a package is sent by a courier to an address given by a customer. The goal of the process is reaching a state where the package is received at the delivery destination and confirmed by the recipient. Now assume the destination address recorded is incorrect (e.g., Birmingham Alabama instead of Birmingham UK). Clearly, it would be impossible for the process to achieve its goal.

While data-aware process design has been investigated to some extent [7][10][9][15][16], the focus of attention has been on combining data flow with activity flow, and avoiding design time errors. A data centric perspective at run time has also emerged [4][5][6][17], allowing changes to the process at run time while maintaining its soundness. However, systems of this kind, like "traditional" process-aware information systems, depend on the quality of the data and build on the assumption that the data is reliable. To the best of our knowledge, avoiding runtime problems that may arise due to data deficiencies in business processes has not been addressed so far. Hence, before solutions can be developed, some understanding of the considered phenomenon should be achieved.

This paper aims at exploring the situation where the information system does not truly reflect the state of a domain where a process takes place, namely, the situation of data inaccuracy [18]. In particular, we ask the following questions: (a) how is data inaccuracy manifested in a process? (b) what are the expected results of data inaccuracy? (c) how can robustness to data inaccuracy be increased? We address these questions using the conceptual framework of the Generic Process Model (GPM) [11][12], which is anchored in an ontology that depicts domain behavior.

The remainder of the paper first provides a formalization of data inaccuracy and some relevant concepts. Then the three questions presented above are addressed. After discussing the questions based on the GPM conceptual framework, we demonstrate the suggested ideas using an example. Finally, conclusions and future work are presented.

2 Formalizing data inaccuracy concepts

This section formalizes the notion of data inaccuracy, based on the Generic Process Model (GPM) [11][12]. GPM is ontologically-based in that it uses a specific set of fundamental constructs developed originally by Bunge [2][3] for modeling "real world" domains. The basic set of concepts includes – things, properties, composition,

attributes, states, events, and laws. A process is considered a "trajectory" of states in a certain domain comprising things. An important aspect is the notion of goal which is a set of states the domain is intended to reach at the end of the process. The left-hand side of Fig. 1 depicts the main concepts used for representing the dynamics of a process in the real world.

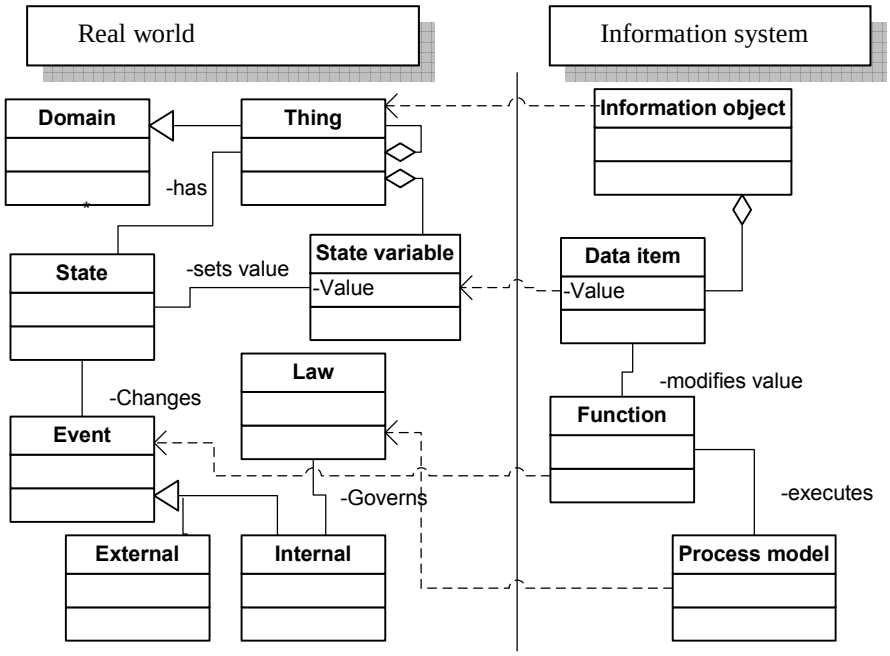

Fig. 1. Real world process and its reflection in an information system

According to GPM, a (real world) process takes place in a *domain*, which is a composite *thing*, including things (e.g., courier, customer) and their interactions. The *state* of a thing (and of the domain) is the set of values of its *state variables* (properties – e.g., address). States are changed by *events* (e.g., package sent), which can be *external* to the domain or *internal*, in which case they are governed by the *law* of the domain. The real world is reflected in the information system, depicted in the right-hand side of Fig. 1, where dotted lines denote a representation relation. Things are represented by *information objects*[1], which have a defined set of attributes or *data items*, whose value reflects the value of a corresponding state variable at a moment in time. Hence, the values of all data items at a moment in time reflect the current state of the domain. Events are reflected by *functions* of the information system (can be some computation or input made by a user), which modify values of data items. Note, these functions can include the creation or deletion of an information object with all

[1] Note, there are also information objects that reflect the history of things (e.g., transaction).

its set of data items. Finally, in systems that include an executable *process model*, it is a reflection of the domain law, according to which the system functions are executed.

The correspondence between a domain and its representation in an information system is formalized in the following definition.

Definition 1: Let $X=\{x_1, x_2,\ldots x_n\}$ be the set of state variables of the domain and $DI=\{d_1, d_2,\ldots d_n\}$ be the set of data items in the information system. $R:X\rightarrow DI$ is a function such that $d_i=R(x_i)$ implies that d_i is the data item that reflects the state variable x_i. We denote the couple $<x_i,d_i>$ where $d_i=R(x_i)$ a *corresponding* couple.

Note that, according to our notation, while x_i stands for a state variable, x_i is its value; similarly, d_i is the value of the data item d_i. As an example, consider the inventory level of an item x_i whose real value at a moment in time is x_i. It is reflected by a corresponding data item d_i in an information system, whose value at that moment is d_i. Changes in the inventory level are reflected as updates in the value of the corresponding data item.

We are interested in exploring situations where the domain state is not truly reflected by the information system state. We use the correspondence relation to define these situations.

Definition 2: A domain state $s=(x_1, x_2,\ldots x_n)$, is truly reflected by an information system state $s_R=(d_1, d_2,\ldots d_n)$ iff $x_i=d_i$ $\forall$ corresponding couple $<x_i,d_i>$. *Inaccuracy of data* is a situation where $\exists$ a corresponding couple $<x_i,d_i>$ such that $x_i\neq d_i$.

Considering the inventory example above, when $x_i\neq d_i$ the information system does not accurately reflect the real inventory level.

Definition 2 is one basis for the following discussion of data inaccuracy and its effects. The second basis is the GPM notion of independent sub-domains which we briefly introduce here (more details can be found in [13][14]).

Definition 3: A *sub-domain* is a part of the domain described by a subset of X.

When looking at a sub-domain at a moment in time, each state variable of the sub-domain has a value, and together they define the state of the sub-domain. This state is actually a *projection* of the state of the entire domain on the sub-domain. The projection of a domain state s over sub-domain Z is denoted as $s_{/Z}$. When the domain transforms and changes its state, a sub-domain can change accordingly. We then say that the domain law is projected on the sub-domain. However, for a given sub-domain, the changes in its state variable values may depend on state variables outside the sub-domain. For example, consider a sales clerk and a sale order as a sub-domain in an order fulfillment process. Accepting or rejecting an order depends on state variables outside this sub-domain, such as inventory levels, production capacity, and the customer's credit. Observing this sub-domain in isolation, the law might seem to be unpredictable.

In some cases, the projection of the law over a sub-domain is such that the transformations depend only on state variables within the sub-domain itself. Consider, for example, a sub-domain of a warehouse receiving goods. The goods may arrive from a supplier or from a sub-contractor, but the projection over the warehouse would not be affected by state variables outside the warehouse itself. In other words, in this

case the law is a function mapping states of the sub-domain to states of the sub-domain. A given unstable state of the sub-domain will always map in the same way, independent of the state of the whole domain, and hence independent of the states of other sub-domains. We will then say that the sub-domain behaves *independently*. Partitioning of the domain into independently-behaving sub-domains is often a consequence of different actors acting in the domain. These actors can be people, departments, machines, computers and combinations of those.

Definition 4: A sub-domain D^1 of D will be called an *independently behaving* (in short an *independent*) sub-domain iff the law projection on D^1 is a function.

Note that a sub-domain can be independent at some sets of states and not at others. Independent behavior of sub-domains is discussed in [13][14] as a necessary condition for concurrency. When sub-domains become independent, they may start operating concurrently; this is represented in process models as a split point. When they reach a state where their (projected) law is not independent, this would be represented as a merge point in a process model. We term the sub-domain which will continue transforming at the merge the *continuation* sub-domain. Various kinds of merge behaviors exist, differing from each other in the conditions that activate the continuation of the process [13]. One known behavior, that can merge sub-domains that operate concurrently, is when the process continuation is activated only when all the independent sub-domains have reached the merge (ceased being independent). This is called a *synchronizing* merge, as formalized in the following definitions.

Definition 5: A set of states S_{sp} is a parallel split iff there exist at least two sub-domains such that for every state in S_{sp} each sub-domain is independent and is in an unstable state.

Definition 6: Let $D^k \subset D$, k=1...n be independent sub-domains operating concurrently following a split point S_{sp}. A set of states S_{me} is a *synchronizing merge* iff a continuation sub-domain D^C exists, such that:

(1) Each sub-domain D^k has at least one unstable state projected to from a domain state for which the continuation sub-domain D^C is stable, and at least one stable state u_k projected to from a domain state where the continuation sub-domain D^C is unstable.

(2) For each sub-domain, there is at least one u_k that projects onto the same unstable state of D^C as all other sub-domains.

(3) There are no other unstable states of the D^C projected into by a state in the merge set S_{me}.

The stability in (1) assures each sub-domain will "wait" for the others before continuation is activated. (2) and (3) assure that D^C will only begin changing when all sub-domains have "arrived" at their "appropriate" states (u_k). Together, these conditions assure synchronization.

To demonstrate the definition, consider a process where several parts need to be manufactured for a product to be assembled. When a production order is given, the domain enters a split state where each part is made by a separate production cell. When each cell has completed making the part, the cell "rests". Only when all cells completed (hence each is at rest – in a stable state), the domain enters a state where the product can be assembled, namely, the continuation sub-domain is activated.

Synchronization of independent sub-domains is a key concept in our discussion of the consequences of data inaccuracy in the following section.

3 How is data inaccuracy manifested in a process?

For discussing the potential consequences of data inaccuracy, let us return to the two scenarios introduced in Section 1. In the first scenario an inaccurate value of material cost was recorded in a quotation preparation process. The process continued and completed without detecting the inaccuracy. In the second scenario, an inaccurate delivery address was recorded in a package delivery process by a courier, preventing the process from achieving its goal and completing.

These two scenarios are clearly different with respect to the effect of inaccuracy. However, the fundamental underlying difference that causes different results of inaccuracy needs to be explored. In particular, we wish to explore the circumstances under which inaccuracy would be acknowledged in a process.

To address this question, we now abandon the separated view introduced earlier of the "real world" domain and its reflection in the information system. Rather, we look at the process domain as including both the real world and the information system. Formally, our process domain will be represented by the set of state variables $X^D = X \cup DI = \{x_1, x_2, \ldots x_n, d_1, d_2, \ldots d_n\}$. Addressing this extended domain, we rely on the notion of independent dub-domains to explain the consequences of data inaccuracy.

In the second scenario above of a package delivered by a courier, the extended domain includes "real" state variables ($x_1, x_2, \ldots x_n$) such as the ordering customer, the delivery destination, etc., and state variables holding their reflections ($d_1, d_2, \ldots d_n$). Once the order details are recorded, the process progresses at a sub-domain which includes some "real" state variables (e.g., means of transportation) as well as reflection data items (e.g., the delivery address). This sub-domain acts independently of the sub-domain holding the real delivery destination. Assuming the delivery address has been incorrectly recorded, the independently acting sub-domain would not be affected by the real delivery address. Relying on the corresponding data item, transportation is arranged, and the package is loaded and sent. At the same time the other (recipient) sub-domain may transform, preparing for the package to arrive. The two sub-domains should cease being independent and synchronize at the merge point, when the package arrives. At the merge point, the domain state is expected to be as follows: the courier sub-domain has brought the package to its destination and became stable; the recipient sub-domain is stable waiting for the package; the process continuation would be triggered when the recipient gets the package. However, due to the mismatch between the real destination and its reflection, the actual state reached is not a state projected to from the merge, since the package is not at the same place as the recipient. Hence, synchronization cannot take place and the process cannot continue.

In contrast, consider the first scenario above where material cost was falsely reflected in a process of quotation preparation. While relying on the reflected value of material cost, the process does not include any step where "synchronization" with a

sub-domain including the actual value is required. Hence, the process may achieve its goal without recognition of the error that has been made.

Summarizing this discussion, data inaccuracy becomes apparent when the sub-domain including the reflected value of a state variable is synchronized with the sub-domain which includes the real value. Such synchronization can be a result of the required course of the process, typically when physical operations are performed.

4 What are the expected results of data inaccuracy?

In the courier example discussed above, data inaccuracy resulted in a situation where the process got stuck and could not achieve its goal. However, this is not necessarily always the case. Different situations might lead to different results, and it is important to be able to characterize and distinguish the cases where the results could be severe.

In the following discussion, summarized in Fig. 2, we assume that the process includes a synchronization point, so inaccuracy can potentially be acknowledged.

One factor that may affect the results of inaccuracy is the existence of multiple paths in the process. There are three possibilities for the law to address a state variable (and its corresponding data item). First, the law may "use" its value on every possible path leading to the process goal. In this case, the value is *mandatory* for the process to complete, and, based on our assumption, synchronization with the real value will be required before or at the process goal.

Second, the law may use its value on a subset of the paths leading to the goal. In this case the value is *optional* for the process completion. If the process takes a path where the value is not used at all, inaccuracy will not affect the process. Otherwise, if the process takes a path where the value is used, once inaccuracy is recognized, it may be possible to roll the process back [8] to a point where a different path can be taken. This might have a negative effect on the business results of the process. For example, if the recipient's cell phone number is discovered to be wrong when delivery needs to be coordinated, some other means of communications may be used (e.g., email). This might take more time than if the number was correct.

Third, once recorded, the value of a state variable might not be used by any transition in the process, and is hence *redundant*. In this case, the value might be needed for other processes in the organization. We may thus assume that analyzing the consequences of its inaccuracy should be done with respect to those processes and not in the analysis scope of the process under consideration.

Even when considering a mandatory variable or one that is used on the chosen process path at runtime, two possibilities exist as to the results of synchronization with the real value. (a) Due to inaccuracy, the process would not reach a state where synchronization is possible. This is the above described case of the courier process. (b) Despite the inaccuracy, the process reaches a merge state, namely, synchronization is possible, and the process can progress towards achieving its goal.

In the latter case, two other possibilities exist. First, it might be that the granularity at which the law operates in indifferent to the inaccuracy in the represented value. For

example, consider an inaccurate value representing inventory level, which should synchronize with the real inventory level when material needs to be issued. The law relates to the amount that needs to be issued. If the real inventory level is above the required amount, then the process can continue and the inaccuracy may not even be acknowledged.

Second, it might be that facing the real value, the process would take a path which is different than the one planned and still achieve its goal. In this case the consequences might be different in terms of business performance measures. In the courier process described above, assume delivery was planned based on a wrong transportation schedule. Once the error is identified, special delivery can still be made, but at higher costs.

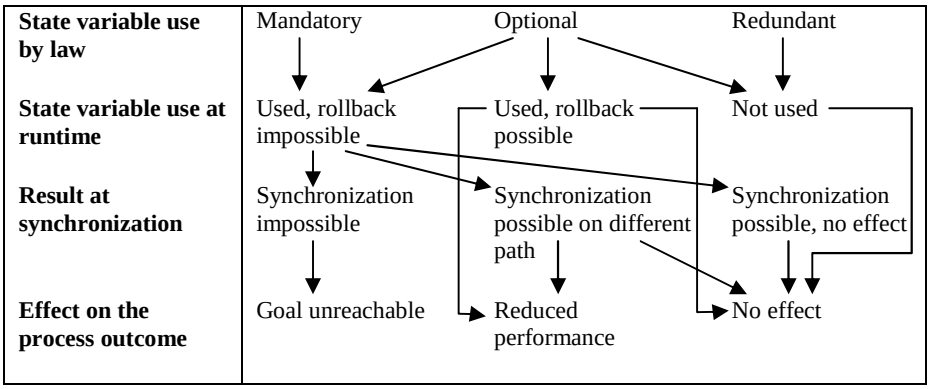

Fig. 2 Possible results of data inaccuracy

5 How can robustness of processes to data inaccuracy be increased?

Data inaccuracy is clearly related to runtime of processes. However, it is possible to take this possibility to account at design time, and design a process to be more robust to errors that may occur in the represented values.

As discussed above, inaccuracy is not necessarily discovered at the moment it occurs. It might be discovered only later on in the process, after many actions have been performed based on an incorrect value, sometimes when it is too late to restore the process to a state from which its goal can be achieved. In other cases, inaccuracy might not even be recognized after the process has reached its goal (e.g., the quotation preparation process), but business results would be harmed.

The following issues come up from our above discussion: (a) Inaccuracy can be discovered when the sub-domain including the reflected value of a state variable is synchronized with the sub-domain which includes the real value. (b) The result of inaccuracy is expected to be different for different state variables that participate in the process.

Considering point (a), such synchronization can be a result of the required course of the process, typically when physical operations are performed. However, it can also be added to the process as a step intended to verify the reflected values, in order to

prevent execution based on false information. In the quotation preparation example, such verification could be achieved by collecting information from different sources (e.g., supplier and inventory records) and comparing the values obtained. It is even not necessary for the verification to use the real value. Rather, verification might be achieved through comparison with similar quotations, or through a review by independent experts. In general, verification would be a step where the process cannot continue unless the values that relate to the IS reflection match some values that are *independent* of that reflection.

Still, we do not propose to introduce verification steps for each state variable value in the process. This would result in inefficiency. Rather, based on issue (b) above, we should identify the state variables whose potential inaccuracy results would be most severe, and make sure their value is verified before any harm is done.

6 Demonstration

This section demonstrates the above presented ideas by applying them to an example process. To visualize the process, we adapt the Workflow nets with Data (WFD-nets) notation proposed by [10][16] for data incorporated workflow models. A WFD-net is a Workflow net, namely, a Petri net with one initial place and one final place, whose transitions are annotated to represent data operations. The notation relates to three operations: write (wt), read (rd), and delete (del). To support our purposes, we add a fourth one, Synchronize (sn), denoting that the real value of a state variable is synchronizing with the process at that transition. Note that WFD-nets also include Guards, which are guarding functions that specify the conditions for selecting a specific path. In our example these are not specified.

Our example process is of organizing outdoor social activities, typically ordered by companies for groups of their employees. When a customer's order is received, the details are agreed upon and recorded. These include the planned date, the location, the type of activity (it can be some sporting activity, a designated workshop, or a guided tour), specific requirements regarding the food (style, number of participants), and required means of transportation. Transportation arrangements are not always needed, and it is possible that the participants will arrive at the meeting location by themselves. On the planned date, physical preparations in the location are made, and the participants arrive (by themselves or by the coordinated transportation). After the social activity is executed, payment is made. The process model is depicted in Fig. 3.

To demonstrate our ideas, we shall consider possible errors in the representation of some of the state variables in the process (each one by itself, not in combination with others).

Planned date – assume the planned date is falsely recorded. The synchronization with the real value of this state variable is on the planned date itself, either when transportation waits for the participants or when the activity itself should take place. Clearly, if the activity is organized for a date which is different than the one the participants prepared for, synchronization cannot take place and the process gets "stuck", unable to achieve its goal.

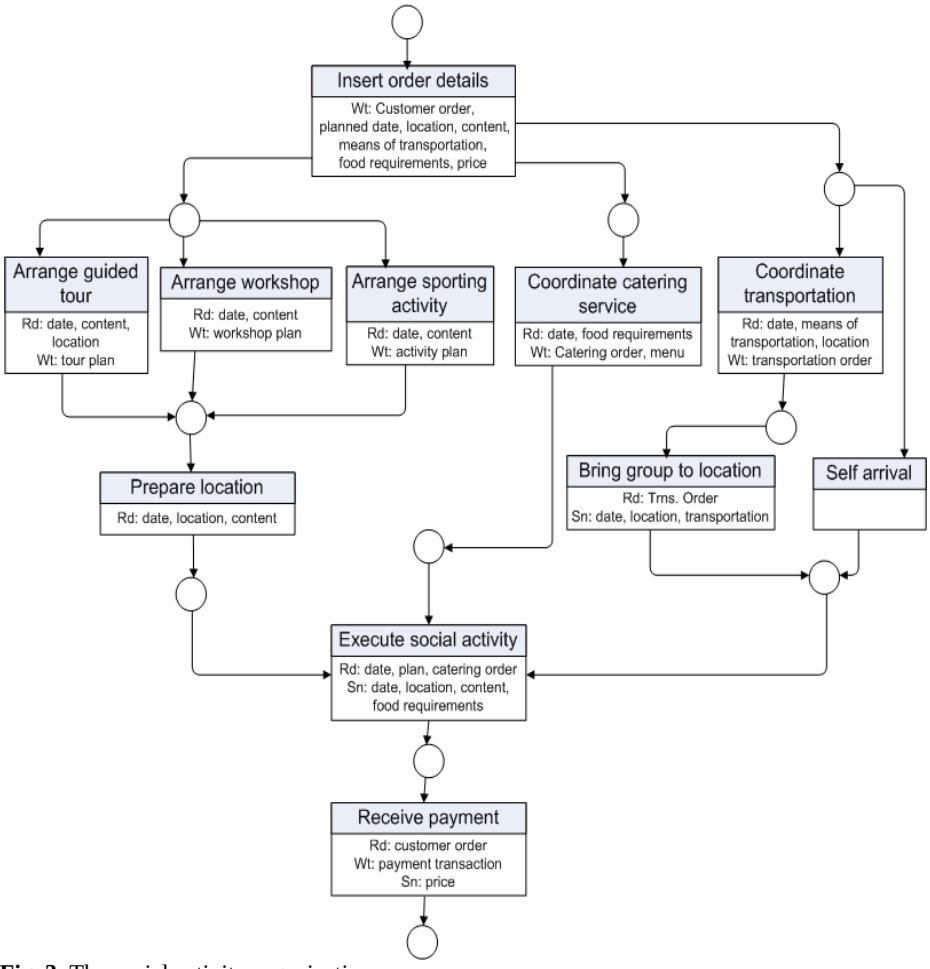

Fig. 3. The social activity organization process

Means of transportation – assume the customer required some means of transportation, while the record in the information system is for self arrival. Synchronization is when the participants would wait for transportation to arrive, which would not take place. However, this path is optional in the process model. Once the problem becomes apparent, it may still be possible to "roll back", so the participants use their own vehicles and arrive at the location. This would take time and reduce their satisfaction, so the process would be able to achieve its goal, but with lower performance indicators.

Food requirements – assume a requirement for vegetarian food was falsely entered. Synchronization of this state variable is while the social activity is being executed, so

it would probably be too late to roll back and prepare other kinds of food. Again, this would not stop the process, but reduce the level of participants' satisfaction.

Price – assume there has been an error in the recorded price. The synchronization of this state variable is when payment is made. Payment would be possible (so the process can still achieve its goal), but profitability might be harmed.

The analysis of inaccuracy with respect to these state variables demonstrates the different possible consequences. Clearly, the most severe would be inaccuracy of the date. Still, inaccuracy would have harmful results for all the other state variables as well. One of the reasons for this is that synchronization only takes place when, in most cases, it is too late to recover without any loss. A possible solution would be to introduce a synchronizing step earlier in the process. For example, it would be possible to send the plans to the customer for approval once they are ready. This would constitute an early synchronization point, and enable detecting inaccuracy when it is still possible to change the plans and avoid any damage to the process.

7 Conclusions

A lot of effort has been devoted to the design of sound process models and the development of process aware information systems to support them. However, these all depend on the quality of the representation in the information system. Humans have learnt to rely on information systems and to make decisions based on the information they provide about the state of the world. Hence, designing business processes to be robust and resilient to deficiencies and inaccuracy in the data stored in the information system is an important challenge.

This paper is a first step towards systematically addressing the possibility of runtime data deficiencies when designing processes. As a first step, it conceptualizes the problem and provides some understanding of its underlying mechanism. It also motivates further investigation of this issue by highlighting the consequences and possible results of data inaccuracy.

As future work, a lot has yet to be done and many questions are still unanswered. In particular this would include identifying the "weak links" – the state variables whose verification is crucial for the process to achieve its goal. Following this, the challenge remains to provide methods that would support the design of processes to be robust and avoid problems related to data inaccuracy.

Acknowledgement: Some of the ideas presented in this paper are the result of discussions with Michael Vaknin, who brought the issue of data inaccuracy to my attention.

References

[1] Agmon, N. and Ahituv, N. , Assessing data reliability in an information system. *Journal of Management Information Systems*. 4(2), pp. 34-41. 1987

[2] Bunge M., Treatise on Basic Philosophy: Vol. 3, Ontology I: The Furniture of the World, Boston, Reidel, 1977.

[3] Bunge M., Treatise on Basic Philosophy: Vol. 4, Ontology II: A World of Systems, Boston, Reidel, 1979.

[4] Cohn D., and Hull R., Business Artifacts: A Data-centric Approach to Modeling Business Operations and Processes. IEEE Data Eng. Bull. 32(3), pp. 3-9, 2009

[5] Künzle V., and Reichert M., Towards Object-aware Process Management Systems: Issues, Challenges, Benefits. Proc. BPMDS'09, pp. 197-210, 2009

[6] Müller D., Reichert M., and Herbst J., A New Paradigm for the Enactment and Dynamic Adaptation of Data-Driven Process Structures. *Advanced Information Systems Engineering (CAiSE'08)* (LNCS 5074), pp. 48-63, 2008

[7] Russell N., ter Hofstede A.H.M., Edmond D., and van der Aalst W.M.P., Workflow Data Patterns:Identification, Representation and Tool Support, In L. Delcambre et al. (Eds.): ER 2005, LNCS 3716, pp. 353–368, Springer-Verlag Berlin, 2005.

[8] Russell, N., van der Aalst, W.M.P., ter Hofstede, A.H.M.: Workflow exception patterns. In: Dubois, E., Pohl, K. (eds.) CAiSE 2006. LNCS, vol. 4001, pp. 288–302. Springer, Heidelberg, 2006

[9] Sadiq S.W., Orlowska M.E., Sadiq W., and Foulger C., Data Flow and Validation in Workflow Modelling. In Fifteenth Australasian Database Conference (ADC), Dunedin, New Zealand, volume 27 of CRPIT, pp. 207-214, 2004.

[10] Sidorova N., Stahl C., and Trcka N., Workflow Soundness Revisited: Checking Correctness in the Presence of Data While Staying Conceptual, Proceedings of CAiSE 2010 (to appear)

[11] Soffer P. and Wand Y., Goal-driven Analysis of Process Model Validity, in: A. Persson, J. Stirna (Eds.), Advanced Information Systems Engineering (CAiSE'04), LNCS 3084, Springer-Verlag, Berlin, pp. 521-535, 2004.

[12] Soffer P. and Wand Y., Goal-driven multi-process analysis, Journal of the Association of Information Systems 8(3), pp. 175-203, 2007.

[13] Soffer P. and Wand Y., and Kaner M., Semantic analysis of flow patterns in business process modeling, in: G. Alonso, P. Dadam, M. Rosemann (Eds), Proceedings of the 5th International Conference Business Process Management (BPM 2007), LNCS 4714, Springer-Verlag, Berlin, pp. 400-407, 2007.

[14] Soffer P., Kaner M. and Wand Y., Assigning Ontology-Based Semantics to Process Models: the Case of Petri Nets, *Advanced Information Systems Engineering (CAiSE'08)* (LNCS 5074), pp. 16-31, 2008.

[15] Sun S.X., Zhao J.L., Nunamaker J.F., and Liu Sheng O.R., Formulating the Data Flow Perspective for Business Process Management. Information Systems Research, 17(4), pp.374-391, 2006.

[16] Trcka. N, van der Aalst W.M.P., and Sidorova N., Data-Flow Anti-Patterns: Discovering Dataflow Errors in Workflows, In: *Advanced Information Systems Engineering (CAiSE 2009)*, (LNCS 5565), Springer-Verlag, Berlin, pp. 425-439, 2009.

[17] Vanderfeesten I., Reijers H. A., and van der Aalst W. M. P., Product-Based Workflow Support: Dynamic Workflow Execution, *Advanced Information Systems Engineering (CAiSE'08)* (LNCS 5074), pp. 571-574, 2008

[18] Wand, Y. and Wang, R. Y., Anchoring data quality dimensions in ontological foundations. *Communications of the ACM* 39, 11, pp. 86-95, 1996