

# Implementing Sentinels in the TARGIT BI Suite

Morten Middelfart<sup>1</sup>, Torben Bach Pedersen<sup>2</sup>

<sup>1</sup>TARGIT A/S

morton@targit.com

<sup>2</sup>Aalborg University – Department of Computer Science

tbp@cs.aau.dk

**Abstract**—This paper describes the implementation of so-called *sentinels* in the TARGIT BI Suite. Sentinels are a novel type of rules that can warn a user if one or more *measure changes* in a multi-dimensional data cube are expected to cause a change to another measure critical to the user. Sentinels notify users based on previous observations, e.g., that revenue might drop within two months if an increase in customer problems combined with a decrease in website traffic is observed. In this paper we show how users, without any prior technical knowledge, can mine and use sentinels in the TARGIT BI Suite. We present in detail how sentinels are mined from data, and how sentinels are scored. We describe in detail how the sentinel mining algorithm is implemented in the TARGIT BI Suite, and show that our implementation is able to discover strong and useful sentinels that could not be found when using sequential pattern mining or correlation techniques. We demonstrate, through extensive experiments, that mining and usage of sentinels is feasible with good performance for the typical users on a real, operational data warehouse.

## I. INTRODUCTION

Bringing data mining to the masses have been a quest by major business intelligence (BI) vendors since the late 1990s [21]. However, a decade later this “Next Big Thing” is yet to happen according to OLAP industry analyst Nigel Pends, author of the OLAP Report [21]. According to the most popular interpretation of Moore’s Law, we can say that within a three year period the computing performance available to an organization will have increased by 400%, during the same period the amounts of structured business data in an organization does not grow more than about 95% (20-30% per year) according to industry analysts IDC<sup>1</sup>. This means that a significant amount of computing capacity is available to analyzing the measures and dimensions in any data warehouse. Therefore, there is an obvious potential to use these computing resources to allow users in an OLAP environment to use data mining for exploring data relationships that are practically impossible to find manually.

We believe that integration of BI disciplines and usability is the key to unlock the big potential of end user data mining that has not yet reached the business users. With this in mind, we have implemented so-called *sentinels* in the commercial TARGIT BI Suite, that is available to more than 269,000 users across more than 3,800 organizations world-wide.

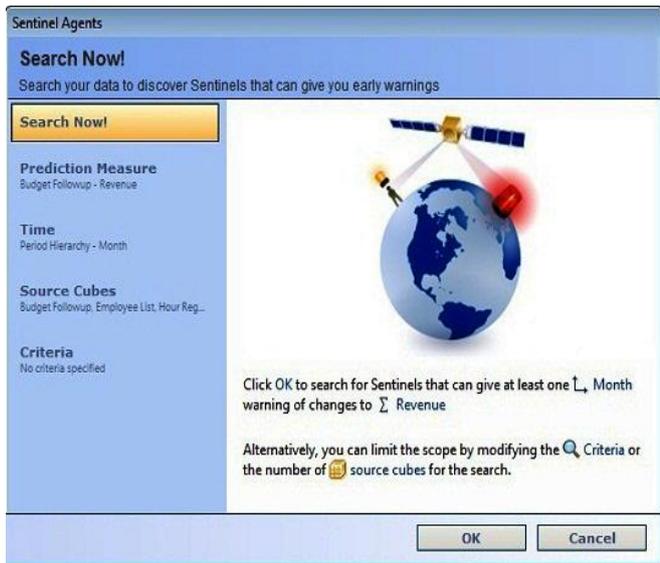
A sentinel is a novel type of causal rule-based relationship; the concept and formal definitions have been developed in

a collaborative research project between TARGIT A/S and Aalborg University [15], [16]. Sentinels are discovered through a data mining process, where changes in one or multiple *source measures* are followed by changes to a *target measure* (typically a KPI), within a given time period, referred to as the *warning period*. An example of a sentinel for a company could be: “IF Number of Customer Problems go up and Website Traffic Volume goes down THEN Revenue goes down within two months AND IF Number of Customer Problems go down and Website Traffic Volume goes up THEN Revenue goes up within two months”. Such a rule will allow a BI system to notify a user to take corrective action once there is an occurrence of, e.g., “Number of Customer Problems go up and Website Traffic Volume goes down”. Using sentinels, the decision maker is now able to respond faster based on a threat to his KPI (Revenue), a threat that might be invisible to him without sentinels. In the TARGIT BI Suite it is now possible for any user to find such sentinels in any given *context*. Subsequently, it is possible to use the sentinels to provide early warnings if critical business goals are threatened. We note that the user only needs to know which measures are important, and thus no data mining knowledge is required.

Sentinels are based on the Computer Aided Leadership & Management (CALM) theory [13]. The idea in CALM is to take the Observation-Orientation-Decision-Action (OODA) loop (originally pioneered by “Top Gun” fighter pilot John Boyd in the 1950s), and integrate BI technologies to drastically increase the speed with which a user “travels” through the OODA loop. Sentinels improve the speed of the OODA loop’s observation and orientation phases by giving the decision maker an early warning (faster observation) that threatens a KPI. At the same time, the sentinel highlights the threat (faster orientation) by listing the measure changes that appears to be “causing” it. In other words, sentinels contribute with both synergy and efficiency for a user cycling an OODA loop.

Sentinels are mined on the measures and dimensions of multiple cubes in an OLAP database, as opposed to the “flat file” formats used by most traditional data mining methods. Sentinels find rules that would be impossible to detect using traditional techniques such as sequential pattern mining and correlation techniques [15]. As explained in detail in Section VI, the *bi-directional* sentinel rules are *stronger* rules than those mined by sequential pattern mining. In addition, sentinel rules are more *specific* than the relationships that can be found using regression techniques.

<sup>1</sup>ComputerWorld, September 22nd, 2009, BI expert, Brian Troelsen, IDC.



(a) Start by selecting the target measure and period for warnings



(b) End by listing the sentinels that can give an early warning

Fig. 1. Searching for sentinels in the TARGIT BI Suite

In the following section, we present the in-depth motivation for sentinels along with a real world case of sentinel application. Section III presents the implementation of sentinels in the TARGIT BI Suite, a commercial software package from the company TARGIT A/S. In Section IV we present our real-world experience with sentinels in the market since their commercial availability in April 2009. In Section V we present experiments of the sentinel discovery process on large amounts of data. Section VI presents the work related related to sentinels, and Section VII presents our conclusions and proposals for future work.

## II. MOTIVATION & CASE

**The TARGIT case:** The data warehouse in TARGIT A/S has been operational for ten years, and it has been continuously adapted to the challenges of the organization. Today, the TARGIT data warehouse is based on a Microsoft (MS) SQL Server 2008 which is used for storage and staging of operational data. Upon transformation into cubes for different business areas, these data are made available to users through the TARGIT BI Suite that resides on top of a number of MS Analysis Services 2008 cubes. The data warehouse contains 16 GB of data, organized in 16 cubes with 250 measures and 109 dimensions. It is “mature” in the sense that there has not been any significant change to number of measures and dimensions over the past few years. The data warehouse covers data from all business areas within TARGIT A/S, i.e., sales, development, support, and administration. Since all the data are integrated, this means that the data warehouse covers a complete life-cycle for all TARGIT customers, e.g., we know how much software they own, how much training they got, how many problems they had, and which suggestions they have for the future versions of the TARGIT software. In addition, the TARGIT data warehouse covers information about the decision process from the showing of interest in TARGIT’s software to becoming a customer. Based on this

information, TARGIT A/S is navigating in global competition through a network of more than 280 partners (resellers/system integrators) world-wide. In this respect, it is also possible for the partners to access part of the data warehouse to optimize their ability to sell and support TARGIT’s software.

A user can access the data warehouse through a Windows based client that connects directly to the TARGIT ANTserver, or through a browser that renders the zero footprint client of the TARGIT NET server that then connects to the TARGIT ANTserver to access the data (See Section III).

The TARGIT case is used in the current section to demonstrate the implementation of sentinels in the TARGIT software. In addition, the “real data” performance studies in Section 3 are based on this case. Finally, whenever we refer to specific numbers of TARGIT customers, users, and partners in this paper, these figures have been extracted from the TARGIT data warehouse on November 2nd 2010.

**A few words about clicks and context:** It is a big challenge to allow users to data mine without any prior technical knowledge. For this purpose we apply “fewest clicks” as a quantitative approach to usability. The rationale is that minimizing the number of interactions (clicks) the user has during the OODA loop equals reducing the amount of training needed as well as the risk of making mistakes; and most importantly, we improve the speed of the cycle. To reduce the number of clicks, we keep track of the user’s *context*. The context includes the measures and the dimension levels and criteria over which they are displayed, as well as the importance of a given measure compared to others, e.g., if a measure is analyzed on more dimension levels or more often than others, then it is most likely important.

The context also allows the user to move very freely between the different BI disciplines, e.g., from a reporting context (characterized by a formalized layout) directly to an analysis context (characterized by displaying the data over more dimensions) with just one click. In other words, the

user can move from the observation to the orientation phase with just one click. Using the context, the user can search for sentinels very intuitively simply by clicking “search for sentinels” whenever he finds something interesting.

**Searching for sentinels – user perspective:** In Figure 1(a) we see the initial dialogue for the sentinel search. The dialog has been launched from an analytical context where the measure revenue was shown over three different dimensions: *Time (Month)*, *Geography*, and *Product*. We note that the system detects that revenue is most “interesting” over monthly periods since this was the context the user was in before. From this point it is possible to initiate the sentinel discovery process or use the dialog to change the Prediction Measure, the Source Cubes, Time, or the Criteria. By proceeding, the search will typically run for a few minutes on the server before Figure 1(b) appears. The run-time is primarily influenced by the number of measures in the data warehouse as seen in the performance study in Section V.

The sentinels in Figure 1(b) were found in the TARGIT data warehouse. The best sentinel is based on a combined relationship between the number of people involved in the decision process for customer projects and the revenue of training courses at the “TARGIT University”. The direction in which the measure changes are related is shown by the red and green (dark and light in grey-scale) bi-directional arrows next to each of the measures. The top sentinel shows that if the number of people involved decrease and the TARGIT university revenue increase, both by 10% or more, then the total revenue for TARGIT A/S is expected to increase by 10% or more within three months. The sentinel is bi-directional and thus works in the opposite direction as well.

By scheduling this sentinel for notification, the user will now be notified with a given frequency, typically equal to the period over which we searched for sentinels (in this case monthly). If the combined incident of “People Involved” increase and “University Revenue” decrease occurs, then the user will receive an email or notification directly on the desktop stating that:

```
Revenue is expected to decrease at least 10% in 3
month(s) because:
- People Involved has increased at least 10%.
- University Revenue has decreased at least 10%.
The prediction has a confidence of 92%.
Click here to TARGIT the notification context,
or click here to review the Agent properties.
```

This means that the user will know three months ahead that something might happen to the overall revenue, and in addition, the user knows which measures to use as context in order to investigate what is causing the problem. At this point we say that the sentinel has contributed with *synergy* in the OODA loop since it alerted the attention very early to a problem that was most likely invisible to the user. In this particular case for TARGIT A/S, it was surprising that the number of people involved in the decision process could be used as an indicator, whereas it has been known for some time that selling more training will typically make a customer

expand his solution. Intuitively, it does however make sense that the more people are involved in a decision process, the more time it will take, and therefore less revenue will be generated on the short-term; and vice versa. In other words, the users are now able to react faster if future revenue is threatened based on this new knowledge.

The TARGIT BI Suite facilitates an even more radical “select all” option, that schedules all sentinels in a “sentinel swarm”. In this case the swarm will be monitoring everything that is going on in and around the organization, and report if something occurs, that seems to threaten a critical measure. Once a warning occurs the user will then decide what to do based on his orientation of the situation. Having a “sentinel swarm”, rather than having only the sentinel rules that makes sense from a human perspective, appears to be an even more synergic approach to facilitating a fast OODA loop.

Once one or more sentinels have been scheduled for notification, and the users start reacting upon these, the confidence of the sentinels will most likely change. The reason is, that the user interferes with the “history” of causality, e.g., if a user successfully addresses the challenge posed by “People Involved” increasing and “University Revenue” decreasing during the next three months, and thereby avoid that “Revenue” decrease, then the confidence of the sentinel itself will decrease. Confidence and other qualitative measures for sentinels are described below. From a user perspective it should be noted that the confidence of a sentinel is fluid, and depending on users seeking to avoid or fulfill the “prediction” of the sentinel, confidence will change based on the users actions. If the confidence of a sentinel changes below certain thresholds (see Section III) set in the TARGIT BI Suite, the user will be notified, but this time the sentinel will suggest that it is removed from the active notifications list. This way the users can manage a fluid set of notifications where newly mined sentinels are scheduled, and irrelevant sentinels are retired.

It should be noted, that a sentinel mining process does not need to be conducted while the user is online. However, as we will see in the experiments in Section V, the sentinel mining process can many times be conducted with a few minutes as the maximum response time. Therefore sentinel mining is also very feasible in an environment where users want to stay online while conducting their mining processes.

**Searching for sentinels – data perspective:** To exemplify the sentinel mining process that takes place, a data example is presented in Table I(a) and (b), where two subsets have been extracted from an example database. Please note that for reasons of confidentiality, the example provided is *not* real data from the TARGIT data warehouse. The source measures have been extracted for January 2008 to April 2009. The target measure has been extracted for April 2008 to July 2009; in other words, for a similar period in length starting three months later. For both source and target measures we have calculated the cases where a measure changes 10% or more, either up (▲) or down (▼), from one month to another. For easy reference, we have assigned short names to the

TABLE I  
RELATIONSHIP BETWEEN SOURCE MEASURES AND A TARGET MEASURE

(a) Source			(b) Target	
Month	<i>PeoInv</i>	<i>UniRev</i>	Month	<i>Rev</i>
2008-Jan	1	115	2008-Apr	900
2008-Feb	2 ▲	115	2008-May	1035 ▲
2008-Mar	2	100 ▼	2008-Jun	1145 ▲
2008-Apr	3 ▲	90 ▼	2008-Jul	950 ▼
2008-May	2 ▼	363 ▲	2008-Aug	1099 ▲
2008-Jun	3 ▲	310 ▼	2008-Sep	1051
2008-Jul	2 ▼	440 ▲	2008-Oct	1188 ▲
2008-Aug	4 ▲	297 ▼	2008-Nov	1021 ▼
2008-Sep	5 ▲	260 ▼	2008-Dec	912 ▼
2008-Oct	6 ▲	230 ▼	2009-Jan	811 ▼
2008-Nov	4 ▼	294 ▲	2009-Feb	1250 ▲
2008-Dec	5 ▲	264 ▼	2009-Mar	1111 ▼
2009-Jan	6 ▲	230 ▼	2009-Apr	986 ▼
2009-Feb	4 ▼	270 ▲	2009-May	1155 ▲
2009-Mar	3 ▼	353 ▲	2009-Jun	1305 ▲
2009-Apr	2 ▼	400 ▲	2009-Jul	1444 ▲

measures as follows: *PeoInv* = “People Involved”, *UniRev* = “University Revenue”, and *Rev* = “Revenue”. As seen in the 16 rows in Table I, the measures *PeoInv* and *UniRev* tend to change in a combined pattern such that when *PeoInv* goes up, *UniRev* goes down, and vice versa. This pattern in the source measures is observed 13 times, out of 15 possible. If we combine this pattern with the subsequent changes to *Rev* three months later, we see that *Rev* changes in the same direction as *UniRev* in 12, out of 13 possible times. Another observation is that the relationship *Rev* and the combination of *PeoInv* and *UniRev* goes in both directions, which is a property we refer to as *bi-directionality*. Intuitively, one can say that if a relationship is bi-directional, then there is a greater chance that the relationship is causal, as opposed to a uni-directional relationship where a pattern is observed for measure changes in one direction only. Consider a case where revenue and staff costs increase over a period of time. This yields the uni-directional relationship that an increase in revenue leads to an increase in staff costs the following month; in this case a decrease in revenue will not necessarily lead to a decrease in staff costs since these costs tend to be more fixed. Therefore, bi-directional sentinels are more desirable. In the example, it is noteworthy that *Rev* changes 6 times up and 6 times down in combination with *PeoInv* and *UniRev* since this “balance” again adds to the likeliness that the relationship is indeed causal. We say that a perfectly balanced sentinel exists in Table I, where changes in *PeoInv* and *UniRev* is able to predict changes three months later in *Rev* with a historical accuracy of 92% (12 out of 13 times).

In addition to the combined relationship of the source measures, we can also observe “simple” sentinels with only one source and one target measure in Table I. However, the *inverted* relationship between *PeoInv* and *Rev*, as well as the relationship between *UniRev* and *Rev*, each have one occurrence (the first two changes) where *Rev* changes in the opposite direction of what we would expect from all other changes. To assess the ability to predict for such sentinels we must first eliminate its internal contradictions. In this case, it

is done by simply deducting the number of times *Rev* changes in the “unexpected” direction from the number of times *Rev* changes in the “expected” direction. This means that both source measures change 14 times, whereas the target measure after elimination changes only 11 times (12 – 1). Therefore the simple sentinels have a poorer historical accuracy of 79% (11 out of 14 times) compared to the sentinel where the source measures were combined. On the other hand, simpler sentinels with fewer source measures have the advantage of being more general than very specific, overfitted sentinels with many source measures, and therefore sentinel simplicity is also important.

**Distinguishing good sentinels from others:** In order to give an intuitive idea of the properties that distinguishes a good sentinel rule from other rules, we provide the following short definition for the quality of a sentinel. The sentinel, denoted by *Source*  $\rightsquigarrow$  *Target*, is based on a set of source measures, *Source*, and a target measure, *Target*, and we have a warning period, *w*, which passes between a change in *Source* to a change in *Target*. In Formulae 1 to 3, *A* is the number of times that *Source* changes in one direction and *Target* subsequently changes in the “expected” direction, minus the number of times where *Target* did not change in the expected direction. *B* is calculated in the same way, but for the changes to *Source* in the opposite direction of *A*. For simplicity, we assume that  $|A| \times |B| > 0$  since we are only interested in *bi-directional sentinels* in this implementation (see explanation for Formula 3 below). *Balance* is used to determine the degree to which a sentinel is uni-directional (*Balance*=0) or completely bi-directional (*Balance*=1), meaning that the sentinel indicates a decrease to the target measure exactly as many times as it indicates an increase. *Conf* tells us how often, when a change in the source measure(s) occurs, the expected change in the target measure subsequently occurs within *w* time.

$$Balance_{Source \rightsquigarrow Target} = \frac{4 \times |A| \times |B|}{(|A| + |B|)^2} \quad (1)$$

$$Conf_{Source \rightsquigarrow Target} = \frac{|A + B|}{\text{Number of changes to } Source} \quad (2)$$

$$Score_{Source \rightsquigarrow Target} = \frac{|A + B|}{\text{Highest } |A + B| \text{ found in sentinels compared}} \times Conf_{Source \rightsquigarrow Target} \times \left( \frac{1}{2} + \frac{Balance_{Source \rightsquigarrow Target}}{2} \right) \times \left( 1 - wp + \frac{(1 + Maxw - w) \times wp}{Maxw} \right) \times \left( \frac{1}{2} + \frac{1 + MaxSource - |Source|}{MaxSource \times 2} \right) \quad (3)$$

Formulae 1 and 2 each represent one quality of a sentinel, but in order to have one single value to determine which sentinel is the best, we introduce *Score* as shown in Formula 3. *Score* takes into consideration the actual number of times there

is support for a sentinel compared to other sentinels in a cube (normalized into  $]0, \dots, 1]$ ), adjusted for contradictions,  $|A + B|$ , as well as the confidence, *Conf*, and *Balance* of the sentinel. It is desirable that a sentinel has a high number of occurrences as well as a high confidence. However, the balance of the sentinel is also important since it allows us to identify bi-directional sentinels, since bi-directional sentinels in general have a higher probability of being causal as opposed to coincidental. In addition, we introduce the threshold, *Maxw*, which is the maximum length of the warning period, *w*, we are willing to accept. The constant, *wp*, represents the warning penalty, i.e., the degree to which we want to penalize sentinels with a higher *w* (0=no penalty, 1=full penalty). The idea of penalizing higher values of *w* is relevant if a pattern is cyclic, e.g., if the indication of a sentinel occurs every 12 months, and the relationship between the indications on the source measure(s) and the target measure is less than 12 months, then the a given sentinel with a warning period *w* is more desirable than the same sentinel with a warning period *w*+12. We also take into consideration that it is desirable to have shorter, general rules, meaning a low cardinality of *Source*. This prevents our implementation from "overfitting" rules and thus generating very specific and therefore irrelevant rules. In other words, Formula 3 will yield a better *Score* when as many as possible of the following conditions apply simultaneously:

- $|A+B|$  is as high as possible, meaning that many changes in the source measures are followed by the "expected" changes in the target measure
- *A* and *B* are equal or close to equal, meaning that a bi-directional sentinel is preferred
- *w* is as short as possible, meaning that sentinels with short warning periods are preferred to sentinels with longer warning periods
- the are as few source measures as possible, meaning that the sentinel is as general as possible

To exemplify, the calculation of *Score* let us review the best sentinel shown in the top on Figure 1(b). The sentinel indicates that Revenue will change at least 10% three months after People Involved and University Revenue have changed at least 10% in accordance with the direction of the arrows. In this case it was found in the data (Table I) that a decrease in People Involved combined with an increase in University Revenue occurred 6 times where Revenue subsequently increased. Moreover, the exact opposite scenario of an increase in People Involved combined with a decrease in University Revenue also occurred 6 times, and here Revenue subsequently decreased. This gives us a completely balanced rule ( $Balance = \frac{4 \times 6 \times 6}{(6+6)^2} = \frac{144}{144} = 100\%$ ) which is a sign of a highly causal relationship between the measures, as opposed to a more coincidental relationship, e.g., if the rule had a tendency to work only in one direction. The changes to the source measures occur one time without the expected impact on Revenue, i.e., the change pattern on the source measures occurs in one or the other direction 13 times in total. This means that the sentinel has a confidence of 92% ( $\frac{6+6}{13} = \frac{12}{13}$ ). Using a maximum of

3 source measures allowed ( $MaxSource = 3$ ), a maximum allowed warning period of 12 months ( $Maxw = 12$ ), and a warning penalty of 0.5 ( $Wp = 0.5$ ) we have a *Score* of 0.60 ( $\frac{12}{14} \times 0.92 \times (\frac{1}{2} + \frac{1}{2}) \times (1 - 0.5 + \frac{(1+12-3) \times 0.5}{12}) \times (\frac{1}{2} + \frac{1+3-2}{3 \times 2})$ ). In the input for this *Score* it should be noted that the highest  $|A + B|$  for any sentinel found at these thresholds was 14.

Using *Score*, we can select the optimal threshold for changes (referred to as  $\alpha$ ) as well as the optimal warning period, *w*. In principle this is done by testing all combinations of  $\alpha$  and *w*, and subsequently selecting the combination where the sentinel with the highest *Score* exist. However, in reality we can disregard some combinations without testing as explained in Section III. In addition to fitting these values optimally, *Score* is used to rank the sentinels found in the output for the user. The sentinels shown in Figure 1(b) have been ordered by their respective *Score*. Furthermore, *Score* also plays a major role in the optimization techniques applied to mine the sentinels efficiently. Finally, *Score* assists the casual users by providing a simple and uniform way to assess the sentinel quality. As mentioned in Section II, it is the primary goal of TARGIT to empower the casual users with powerful analytics. However, for more advanced users, it is possible to toggle the default constants in *Score* and the thresholds for *Balance* and *Conf*.

### III. IMPLEMENTATION

**Architecture of the TARGIT BI Suite:** The TARGIT BI Suite is a client/server based application which allows users to create reports, analysis, and dashboards (scorecards). In

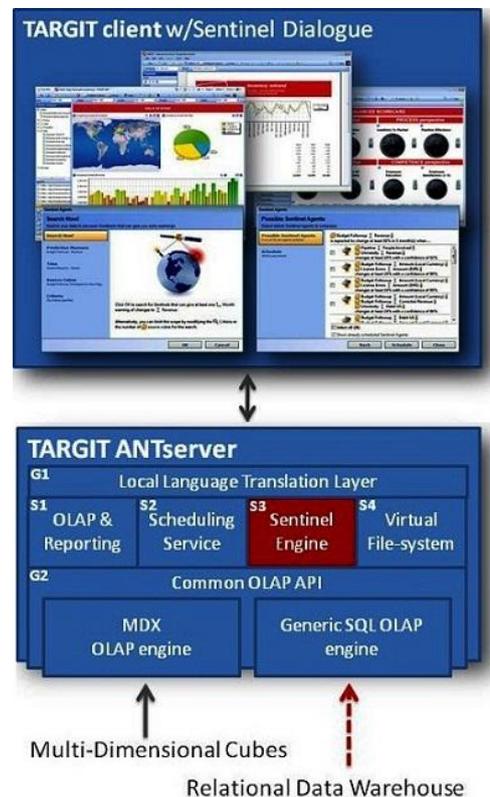


Fig. 2. Architecture of the TARGIT BI Suite

addition, the application allows users to search for sentinels and to schedule these for early warnings as explained in Section II. The user connects a Windows based client program to a centralized TARGIT ANTserver through TCP/IP as shown in Figure 2. The TARGIT ANTserver has a number of services that can be offered to the client; a core functionality of these is to parse and forward a data query to either a cube or a relational database, and subsequently return the answer to the client. An alternative web client is also available. In this case the browser will render a “zero footprint” instance from a TARGIT NET server which holds the execution of the client. The TARGIT NET server will then connect to the TARGIT ANTserver similarly to the stand alone client shown in Figure 2. The TARGIT ANTserver is the backbone of the application. As shown in Figure 2, there are two general services (G1,G2) and four specific services (S1...S4) offered by the TARGIT ANTserver:

*Local Language Translation Layer (G1)* translates dimension and measure names as well as names for files shared among users. This layer also provides the language for the client application. Since the client program receives all language settings from the TARGIT ANTserver based on the user preference means that different users can access the same client program on a physical PC in different languages.

*Common OLAP API (G2)*: This service facilitates that the functionality of the TARGIT ANTserver exceeds traditional MDX and SQL, e.g., the TARGIT ANTserver can run queries across multiple heterogeneous data sources and combine the answers. In addition, it has extended capabilities for handling time-dimensions dynamically, e.g., it allows criteria such as “time must be equal to this month” and other generic time variants, e.g., This year, previous year, previous month. Most importantly, the Common OLAP API allows all other services to treat the entire set of data sources as one homogenous cube with one unified interface.

*OLAP & Reporting (S1)*: This service parses the incoming analytical and reporting queries in XML format from the client and submits them to the Common OLAP API. The results may require post-processing for additional formatting as stated in the XML, and upon completion the result is submitted in binary format to the client. The binary format for the response has been selected to optimize performance.

*Scheduling Service (S2)*: This service facilitates that some jobs can be stored and executed at certain times. The jobs include reports and analysis as well as more general “slideshows” and “podcasts” of the information from the databases. The service also allows agents that monitor specific information to be scheduled, and with a predefined frequency the ANTserver will verify if the premise for notification of the user is met. If so, the user will be notified via email or directly via a special desktop notification client. One type of such agents are the sentinel agents that notify the users if the premise of source measure changes are met.

*Sentinel Engine (S3)*: This new service facilitates the search for sentinels once the client dialogue on Figure 1(a) is completed. From this point, the *SentHiRPG* algorithm,

described below, performs the search for sentinels. Once the search is completed, the answer is returned to the client as seen in Figure 1(b), where it is possible to submit the sentinels found to the scheduling service (S2).

*Virtual File-System (S4)*: This service allows users to store files (reports, analysis, and dashboards) in a repository where they can be shared with other users. The service also facilitates that Windows client files are available on the zero footprint web client through the TARGIT NET server.

**The SentHiRPG Algorithm:** The *SentHiRPG* algorithm that can find so-called generalized sentinel rules was first presented in [16]. In order for this paper to be self-contained, the following is a brief summary of *SentHiRPG*.

*SentHiRPG* applies a novel *Reduced Pattern Growth* (RPG) optimization that quickly identify which measures that are candidates for the strongest relationships. RPG is facilitated by an intermediate optimized format called *The Table of Combinations* (TC). In addition, *SentHiRPG* applies a *hill-climbing* approach to fit the best sentinel warning period.

*The Table of Combinations* is an intermediate hash table that is generated in one pass over the input data, and used for optimization. Once generated, the TC represents exactly the measure change combinations needed to mine all potential sentinels. A reliable sentinel will require at least two, but typically multiple rows in the TC, and since we do not know which source measure changes that occur at the same time, there is no generic sorting method that can optimize the scans further than the reduction of data in the TC.

*Reduced Pattern Growth* delivers a good approximation of the top sentinel rules, and it is much more efficient than a full pattern growth of all combinations of source measures. The idea is to first very quickly identify the source measures that change whenever the target measure change; at this stage we do not attempt to find out what type of relationship the source measures have to the target measure or to each other. For example, if we look at Table I, we see that both “People Involved” and “University Revenue” have 13 occurrences where they change while “Revenue” also change. In this case we say that both these source measures have an *influence* of 13, and we note that there is no guarantee that a high influence will lead to a high *Confidence*, *Balance*, or *Score*. However, a low influence would most likely mean a low *Confidence* and subsequently a low *Score*, since the given source measure would then change very little in combination with the target measure. Once the influence for all source measures have been calculated, they are ranked, and a Pareto Principle is applied to select the source measures that account for a certain percentage (*RPGpareto*) of the sum of all influences. For example, let us assume that we have two additional source measures, “Staff Salaries”, representing the salaries paid to employees, and “Product Quality”, representing the number of errors found in the products sold. The source measure “Staff Salaries” has an influence of 2, and the source measure “Product Quality” has an influence of 3. The sum of all influences for the four source measures

is then  $13 + 13 + 3 + 2 = 31$  (note that the influences are ordered descendingly). If the value of  $RPG_{pareto}$  is set to 85%, we need to identify the most influential source measures that account at least 85% of all influence. Starting with the highest influence observed, 13, we find that the two source measures, “People Involved” and “University Revenue” account for  $\frac{13+13}{31} = 84\% < RPG_{pareto}$ . In this case we need to add the influence of the second highest influential source measure to account for at least 85% of all influence,  $\frac{13+13+3}{31} = 94\% \geq RPG_{pareto}$ . In this example, the process would only eliminate one out of four source measures. However, from experiments on real-world data, we know that the influence of source measures can be described as a *power law*, meaning that a few source measures have a high influence and are thus very likely to be part of many good sentinels, whereas the majority of source measures are not likely to be part of any good rule at all.

Having identified the most influential source measures, we grow sentinels from these measures. Starting with 1 source measure, we add the remaining influential source measures one at a time to create longer rules until the number of source measures equals  $MaxSource$ . During this process we only store a sentinel, and continue to add source measures, if the added source measures translates into a higher *Score*. For example, using the data in Table I we would first create two sentinels with only one source measure, respectively “People Involved” and “University Revenue”. Using the calculation demonstrated in Section II, both these sentinels have a *Score* of 0.56 ( $\frac{11}{14} \times 0.79 \times (\frac{1}{2} + \frac{0.99}{2}) \times (1 - 0.5 + \frac{(1+12-3) \times 0.5}{12}) \times (\frac{1}{2} + \frac{1+3-1}{3 \times 2})$ ). If we combine the two source measures into a longer rule, we have a *Score* of 0.60, and thus the new longer rule will “survive”. Subsequently, we would combine this sentinel with the remaining influential source measure, “Product Quality”, to see if *Score* improved even more, and so on.

On a realistic dataset, the quality of the RPG approximation has been experimentally tested to produce 100% of the top 10 sentinels, and 88% of the top 100 sentinels that would otherwise have to be found by a full pattern growth, i.e., testing all combinations of source measure relationships with the target measure. The performance cost of RPG is only 14% of the comparable cost for a full pattern growth.

*Hill Climbing* is used to auto-fit of the warning period. This is done by identifying the warning period,  $w$ , where the sentinel with the highest *Score* exists. Alternatively, all *Scores* for all sentinels for a each value of  $w$  would have to be inspected to find the sentinel with maximum *Score*. Using a specialized 2-step/1-step hill climbing with two starting points, this approach only consumed 53% of the time compared to testing all possible warning periods.

The SentHiRPG algorithm can now be described as three steps that incorporate these three optimizations:

- Step 1:** Build TC during one scan of the input data.
- Step 2:** Hill climb  $w$  to  $\max(Score)$  of sentinels constructed from TC with source measures found in **RPG**.
- Step 3:** Output sentinels for  $w$  that meet the quality thresholds.

**Implementation of Sentinel Mining:** The implementation of sentinel mining in the TARGIT BI Suite consists of two parts: 1. The dialogue shown in Figures 1(a) and 1(b) which has been implemented in the TARGIT client, and 2. the sentinel engine in the TARGIT ANTserver (S3 in Figure 2). The client dialogue allows the user to manipulate the input parameters before sending the “sentinel mining query” to the sentinel engine. Upon completion of the mining process, the sentinel engine will transfer the sentinels found to the dialogue presenting them. From this stage the user can select one or more sentinels to become agents and submitted to the TARGIT ANTserver’s scheduling service (S2 in Figure 2). Since the dialogue on the client side is a simple matter of manipulating parameters and presenting output, we will focus on the implementation of the sentinel engine in the ANTserver.

The process implemented in the sentinel engine is shown in pseudo code as the *Sentinel Mining* function below. The declaration of the interface for the *SentHiRPG* function is strictly aligned with the definition in [16]. With this prerequisite, sentinel mining can be described in three steps as follows:

**Step 1:** We use the properties of the Common OLAP API to find all measures from all the cubes in *CubeSet* selected by the user. In this context we note that a relational database will also appear to be a cube in this context since it will have been mapped into dimensions and measures when it was plugged into the Common OLAP API by an administrator of the data warehouse. The time-dimension,  $T$ , is a shared dimension that allows the Common OLAP API to relate the measures collected to each other. The dialogue in the client

**Function: Sentinel Mining**

**Input:** A target measure,  $TM$ , a shared time-dimension level,  $T$ , a set of cubes, *CubeSet*, a set of criteria on dimension members, *SliceCriteria*, and a max number of sentinels to be returned,  $X$ .

**Output:** Sentinels with a given warning period,  $w$ , a threshold for indications,  $\alpha$ , and their respective *Conf* and *Score*.

**Method:** The sentinels are mined as follows:

**Sub-Function: SentHiRPG**

**Input:** A list of facts from a cube,  $C$ , ordered by  $(d_2, d_3, \dots, d_n, t)$ , an offset,  $o$ , a maximum warning period length,  $Maxw$ , a maximum number of source measures per rule,  $MaxSource$ , a warning penalty,  $wp$ , a threshold for RPG,  $RPG_{pareto}$ , a threshold for indications,  $\alpha$ , a minimum *SentSupp* threshold,  $\sigma$ , a minimum *Conf* threshold,  $\gamma$ , and a minimum *Balance* threshold,  $\beta$ .

**Output:** Sentinel rules with a given warning period,  $Optimalw$ , and their respective *SentSupp*, *Conf*, *Balance*, and *Score*.

**Method:** The algorithm is described in previous section above.

**Step 1:** Using the Common OLAP API, all “pure” measures from the cubes  $\in CubeSet$  are identified as source measures,  $SM$ , with the exception of  $TM$ . The facts to be mined,  $C$ , are extracted with one dimension (level),  $T$ , and the measures  $SM$  and  $TM$  for all  $T < current\ period\ of\ T$ . If  $SliceCriteria \neq \emptyset$  then each element is applied as a selection criteria to the cubes where it is possible.  $Maxw$  is set to the maximum cardinality of the children sets at the same level as  $T$ . The remaining parameters are set as follows:  $o = 1$ ,  $MaxSource = 3$ ,  $wp = 0.5$ ,  $RPG_{pareto} = 85\%$ ,  $\sigma = 5$ ,  $\gamma = 80\%$ ,  $\beta = 80\%$ , and  $X = 200$ . These settings are based on practical experiences with real world data.

**Step 2:** Repeat *SentHiRPG* for each  $\alpha \in \{10\%, 15\%, 20\%\}$ , all other parameters remain constant as set in step 1. While testing different values of  $\alpha$ , the set of sentinels for a given value of  $\alpha$ , that contains the sentinel with the highest *Score* is stored in memory. This set is not flushed from memory until a sentinel with a higher *Score* exists in another set for a new value of  $\alpha$ .

**Step 3:** Output the top  $X$  sentinels from memory to client through the Local Language Translation Layer. If the number of sentinels  $< X$  then return all.

will only present cubes for selection that include this shared time-dimension. During the identification of all measures we seek to disregard measures that are only replicas of other *base* measures. If for instance a measure is calculated based on only one other measure, we will disregard the calculated measure and stick with the “base” measure which it was based on. Logically we will also disregard measures calculated on measures that are disregarded. By disregarding measures that are not base, we seek to eliminate measures that do not contribute with something new, and thereby we reduce the number of measures that needs to be mined. In other words, we eliminate the most trivial of relationships prior to the mining process, e.g., the relationship between revenue, cost, and profit.

We apply one general criteria on the time-dimension,  $T$ , that seeks to ensure that we do not mine on an incomplete period. This criteria means that we will only mine all periods prior to the one we are in, e.g., if we are currently on November 15th, we will only be mining all data up until and including October which is the last complete month. The same principle applies to whatever period we can think of, e.g., hour, week, or year. In addition, we apply the additional slicing from *SliceCriteria* if such has been specified. However, in this context we only apply these criteria where it is possible since the slicing members do not need to be from shared dimensions. This means that only cubes where it is possible to say “data must be from United States only” gets this criteria applied. One example of this could be that revenue can be selected for United States only, whereas the currency rate of Euros cannot. Nevertheless, there might be an interesting relationship between the revenue in United States and the Euro rate anyways. Therefore we are more loose in applying these criteria than with the strict shared time-dimension which is needed to relate and stage data for Step 2.

Setting  $Maxw$  to the maximum cardinality found in the children sets at the same level as  $T$  means that  $Maxw$  will never exceed the number of periods that is possible for a parent in the hierarchy of  $T$ . If  $T$  is on month level and we are currently in November, and a complete previous year of data exists, then  $Maxw = 12$ . If no previous year exists then  $Maxw = 11$ . In general, this means, e.g., that if we are mining on months,  $Maxw \leq 12$ , if we are mining on weeks,  $Maxw \leq 53$ , and if we are mining on minutes,  $Maxw \leq 60$ . We take for granted that a time-dimension is constructed such that this is possible since that is the implementation “best practice” of TARGIT. The reason for limiting  $Maxw$  not to exceed the timeframe of a parent is, that we only want to find the sentinel for warning period,  $w$ , and not for warning periods  $= w + (\text{parent timeframe} \times n) | n \in \mathbb{N}$ . This would be the case if a sentinel is cyclic over time, and we did not limit  $Maxw$ . For cyclic sentinels we prefer the shortest warning period,  $w$ , since we attribute greater quality to sentinels based on the “latest intelligence from the operational environment”, as opposed to a long-term prediction pattern. In addition, a shorter warning period means that more cycles in the OODA loop (and thus faster adaption) are possible based on the sentinel.

The remaining parameters for  $o$ ,  $MaxSource$ ,  $wp$ ,

$RPGpareto$ ,  $\sigma$ ,  $\gamma$ ,  $\beta$ , and  $X$  are silently applied by the system for most users. It is possible for expert users to change a registration file to modify these. However, the “few click” ambitions in the TARGIT BI Suite suggests that the system makes a “best bet” choice on behalf of casual users. The parameters have been selected based on experiments with sentinel mining on real world data from TARGIT A/S, NASDAQ OMX stock exchange, and the governmental institute Danish Statistics.

*Step 2:* In this step, the *SentHiRPG* function is called with three different values of  $\alpha$ . The three set of sentinels mined competes with the maximum *Score* found in the set, and the set that “wins” is kept in memory. The reasons for testing specifically these three values of  $\alpha$  is rooted in the same real world experiences as stated under Step 1.

*Step 3:* This step returns the  $X$  best sentinels found along with the length of the warning period,  $w$ , and the value of  $\alpha$ . The sentinels found passes through the Local Language Translation Layer in order to be localized for the user. In practice this means that all the measures that constitute the sentinels are translated to the user’s language. The reason for having  $X$  is to limit the load on the visual interface of the client dialogue. It was found that a stress load of sentinels to the client would overload the visual control, and since several hundreds (or thousands) of rules would not contribute meaningfully to the users overview anyways, a threshold was set to 200. Based on real world experiences this threshold is rarely exceeded, and thus most often all sentinels are returned to the client.

**Computational Complexity of Sentinel Mining:** The computational complexity for our implementation is  $\mathcal{O}(c \times p(85\%)^3 \times k^3 \times m)$ , where  $c$  is the number of unique facts in  $C$  (which in this implementation is the size of  $T$ ),  $p$  is the percentage of remaining source measures after RPG optimization expressed as a function of  $RPGpareto = 85\%$ ,  $k$  is the number of source measures ( $MaxSource = 3$ ), and  $m$  is  $Maxw$ . Thus, we will expect linear behavior when scaling the size of  $T$ . The length of the period,  $Maxw(m)$ , in which we fit the warning period is also expected to scale linearly. When scaling the number of source measures ( $k$ ) we will expect the running time to rise cubically without the RPG optimization. However, since the effect of the RPG optimization ( $p(85\%)$ ) is also cubic we will expect this to be an efficient countermeasure to the impact of scaling the number of source measures (see Section V).

**Using sentinels after discovery:** Upon completion of the Sentinel Mining process, the sentinels found will be listed in a user dialogue as shown in Figure 1(b). The dialogue allows the user to select any number of the sentinels found, and to schedule these for notification at a given frequency. The frequency defaults to the same period as the hierarchical level of  $T$ , since it rarely makes sense to test for the existence of the premise for a sentinel on a shorter period than it was mined for. Upon scheduling, the sentinel is basically a traditional *agent*, that tests with the frequency set for the existence of the premise for the sentinel in the most recent period of  $T$  at the level for which it was mined. If the premise for the sentinel is found, the user is notified by email, directly on his computer desktop, or iPhone with a message as shown in Section II.

As mentioned in Section II, the confidence of a sentinel is fluid as time progresses since the user's actions will interfere with the "predictive power" of the sentinel. In fact, all the quality measures are variable as time progresses. However, we choose not to reconsider *Balance* and *Score* since the causality and ranking was only important when the user had to identify one or more relevant sentinels among others. Now that the user has deemed a given sentinel relevant, there is no reason to reconsider the relevancy quantitatively. There is of course the option that a user can unschedule a sentinel manually at any time if he does not find it relevant anymore.

The sentinel prediction and confidence is shown in a notification to the user, if confidence meets the threshold. If the confidence of a sentinel tested falls short of the threshold, the user is notified that the sentinel is recommended to be *retired* as described in Section II. The retirement of a sentinel from active notification concludes the life-cycle of a sentinel.

The cost of checking for the premises for all sentinels is linear in the number of sentinels times the number of source measures per sentinel, since each check requires reading two cells for each included source measure. We recall that the cube (per recommendation) is aggregated on  $T$ , and thus each aggregated cell can be accessed through a hash-based lookup [6]. Thus, the cost of "sentinel check" is much lower than the cost of mining sentinels, and can typically be performed in a few seconds. This assumption is verified in Section V.

#### IV. MARKET EXPERIENCES

Sentinels was launched as a new feature in the TARGIT BI Suite version 2K9 which was released to the market in April 2009. From version 2K9 SR1 (service release 1), released in November 2009, and forward, the implementation is done as described in this paper. Since most sentinels are likely to be based on business data where a weekly or monthly warning period is desired, time has simply not permitted more experiences with sentinel usage than the internal experiences in TARGIT A/S. However, initial feedback on sentinel mining from partners representing a market footprint of 1,936 customers with more than 124,000 users has been positive.

In addition to TARGIT's own surveys, leading industry analyst Gartner has recently accepted TARGIT into the so-called Magic Quadrant for Business Intelligence Platforms in 2010 [22]. The sentinel technology is specifically listed by Gartner as one of the unique key strengths of the TARGIT.

Even though the business impact of sentinels is expected, but not yet to be seen in the large scale, we have learnt a few things about the user interaction with the sentinel mining implementation. With regards to the dialogue where the sentinels found are presented, Figure 1(b), it has been desired by users to see the *Score* for each sentinel as a number. In the current implementation only confidence is presented as a number, whereas *Score* is discretely presented as a sort order where the user knows that the best sentinels are on top of the list. In addition, it has been desired by users to have easy access to a visualization of the relationship between the source measures and target measure. This can be done by visualizing

the measures that constitute the sentinel as bar-charts alongside with the changes highlighted (color-coded). This should of course be available with a "single click".

#### V. EXPERIMENTS

**Setup:** We run our experiments directly on the data of the live TARGIT data warehouse described in Section II. Our setup is implemented across two physical servers:

- 1) a Quad Core Intel Xeon X3220 2.4 GHz with 8 GB RAM, 2 x 73 GB SAS harddrives in RAID 1, running MS SQL Server 2008 cubes on a Windows 2008 64-bit operating systems.
- 2) an Intel Core2 Quad CPU (Q6600) 2.40GHz PC with 4GB RAM and 2 500GB disks (7,200 RPM) running the TARGIT ANTserver on a 64Bit version of Windows 2003 Server R2, Service Pack 2.

We run on version 2K10 of the TARGIT BI Suite (currently in development) that is expected to launch in April 2010. A client is accessing the TARGIT ANTserver, and this client has the following specification: Intel Core2 Dual CPU (6400) 2.13GHz PC with 2GB RAM and 1 250GB disks (7,200 RPM) running a 32Bit version Windows 7. However, it should be noted that the client did not do any processing of the data, it simply relayed a "sentinel mining" or "sentinel check" query to the TARGIT ANTserver on data warehouse server #2 and waited for a reply. Upon reply, the time consumed was recorded. The experiments shown in Figure 3 and 3(d) represent the average of 5 runs each, and the sequence of all the experiments is randomized. To run queries on 10, 20, 30, ..., 140 source measures, we produce descendants of the full data warehouse by selecting a given number of source measures randomly. Please note that the time-scale on Figure 3(c) is different from all other figures.

In Section III we describe how we reduce the number of measures by considering only "base" measures in the mining process. In addition, we naturally only mine the measures that have a relationship (a shared time-dimension). Whenever we refer to a number of measures in the following experiments, we refer to the number of related base measures, e.g., in the TARGIT case there are 250 measures in total, but the largest possible number of related base measures is 154. The TARGIT data warehouse contains data on all related base measures from January 2002 and forward, meaning that we have 8 complete years of data, 2002 to 2009, to run our experiments on. We note that even though a company has significantly larger datasets in terms of transactions, the work for the algorithm would be the same since it is running on the aggregated data in the data warehouse cubes. Thus, it is solemnly the size of the time dimension combined with the number of measures that determine the workload for the algorithm.

**Scaling Periods:** In Figure 3(a) we scale the size of the input, i.e., the number of periods over which we search for sentinels, on 50 measures (49 source measures and 1 target measure), and with a fitting of the warning period over 7 days (1 week, the duration of the parent level of the time dimension). To get the largest possible dataset, we run the sentinel mining on the

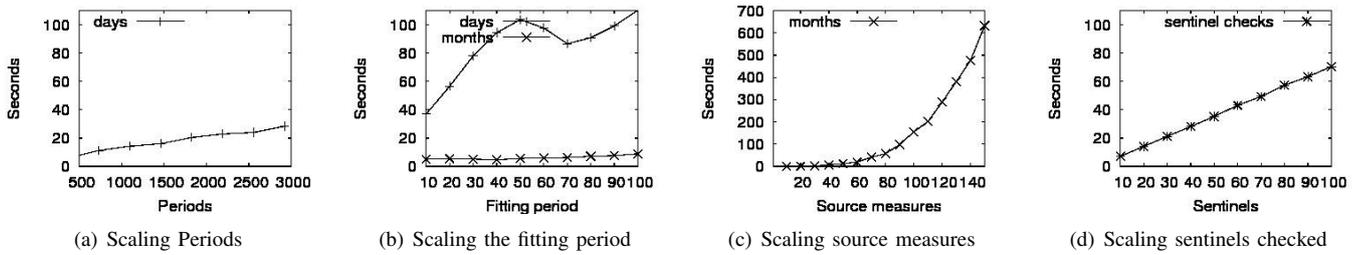


Fig. 3. Sentinel Mining Experiments

“days” level of the time hierarchy which gives us a maximum of 2,922 periods for 8 years. We start by mining on data from 2009 which accounts for 365 days. From this point, we add one year at the time to increase the number of periods with either 365 or 366 days depending on the year. The results are presented in Figure 3(a), and we note that the sentinel search scales linearly from 365 to 2,922 periods as expected based on our assessment of the computational complexity. We should also note that 2,922 periods in this context is a *huge* dataset, since users would normally be operating on higher aggregated levels such as weeks or months. However, even at this finest granularity in the TARGIT data warehouse, the linear scalability seems to more than adequate to facilitate real world usage on a broad user scale, since the mining on even this finest granularity can be conducted for the individual user in less than 30 seconds (we recall that the cube data is pre-aggregated at the period(s) and level(s)).

**Scaling the fitting period:** In order to scale the time frame, in which we fit the warning period, as seen in Figure 3(b), we needed to tweak the default settings in the TARGIT ANTserver to omit the built in prevention method for finding the same cyclic sentinel multiple times as described in Section III. In this experiment, we mine sentinels on 50 measures (49 source measures and 1 target measure) on “days” and “months” granularity respectively, and we scale the “fitting period” from 10 to 100 periods. Based on our assessment of computational complexity, we would expect both levels on the time dimension to scale linearly. We do indeed observe a linear behavior for “days” and “months” when the fitting period is within the default settings, i.e., “days” will default to a fitting period of either 7 or 31 periods (days), and “months” would default to a fitting period of 12 periods (months). However, when scaling beyond the default settings only the scaling on the “months” level displayed a linear behavior. The “days” level seemed linear in behavior from 10 to 50 periods, but then dropped in time consumption from 50 to 70 periods, from where it again displayed linear behavior. The reason is, that another local maximum for *Score* exists close to a warning period of 70 which has better score than a local maximum between 1 and 10. This means that our hill climbing part of the SentHiRPG algorithm will have a tendency to be attracted towards the longer warning periods once “fitting period”  $\geq 60$ . Such a signature suggests that the sentinels mined on “days” are cyclic on a bi-monthly basis. However, as explained in Section III, we maintain that a shorter warning period is usually more relevant and actionable for a user than a longer period.

Another interesting finding in Figure 3(b) is that, although the mining on “days” is linear in two period-sequences, it seems to be much more costly than mining sentinels on “months” when scaling the “fitting period”. When investigating this, we found that the sentinel mining only finds two sentinel on the “days” level that actually meets the thresholds, meaning that all the time is used to process a lot of very similar, but very poor sentinels. At the moment, the RPG optimization benefits from the existence of influential measures in the input data, but if a lot of poor measures, and no good measures, are input then the effect of the RPG optimization is low. Effectively, this means that we spend *more* time finding *less* and *poorer* sentinels. We should note that even though the sentinel mining is not used on an optimal dimension level from neither a performance nor quality perspective, the performance of the system is still fair, since even the longest running query takes less than two minutes. This is more than adequate for all users, since the user does not have to be online while conducting the sentinel mining. In this context, we also note that when running on the “month”, which produces more useful sentinels from a business perspective, the longest running mining process takes less than 9 seconds.

**Scaling source measures:** In Figure 3(c) we scale the number of source measures from 10 to 150 on a realistic sentinel mining process at the “months” level of the time hierarchy. During this process the warning period is fitted over 1 year. We notice that increasing the number of source measures is expected to have a cubic impact based on our assessment of the computational complexity, and we have verified in the statistical tool R ([www.r-project.org](http://www.r-project.org)) that the curve in Figure 3(c) is indeed cubic. Although the curve seems steep, we should note that the impact of scaling the source measures would be more than seven times higher without the effect of the RPG optimization. We attribute the efficiency of the RPG optimization on real data to the existence of a power law in the data, where a few source measures have strong relationships with the target measure whereas a larger number of source measures has little or no relationship with it. From a user perspective the performance is adequate, since the longest sentinel mining process possible in the TARGIT data warehouse at this level took 10.5 minutes, and as we recall, the user does not need to be online while mining. Moreover, the wait is worthwhile since the quality of the sentinels found will benefit the user with early warnings for months to come. We recall that once mined, the sentinel can be scheduled and “checked” (whether to warn or not) with a given frequency,

and the time consumption of a sentinel check is less than a second in a real, operational data warehouse.

**Scaling sentinels checked:** In Figure 3(d) we scale the number of sentinels for which we check for the existence of their respective premises. We note a complete linear scalability when scaling to a realistic number of sentinels for a reasonably large "sentinel swarm" (see Section II). This is consistent with what we would expect based on our assessment of computational complexity of such a process (see Section III). We recall that our sentinel mining returned 26 possible sentinels for *Revenue* in Figure 1(b), and thus 100 sentinels is roughly equal to scheduling four sentinel swarms to guard four critical measures. The cost of checking the entire number of sentinels requires just about one minute of processing (precisely 1 minute and 10.5 seconds), which means that a monthly follow-up on a sentinel swarm is realistic for a large group of users.

**Experiments Summary:** In summary we have demonstrated that the sentinel mining in the TARGIT BI Suite (with default settings) scales linear in number of periods and length of fitting period, and it scales cubically in number of source measures. We confirmed that the current settings in the TARGIT BI Suite are adequate to prevent users from mining cyclic sentinels, and that these settings will also prevent users from spending excessive time and system resources in doing so. Most importantly, we have demonstrated that both mining and usage of sentinels is feasible with good performance for the typical users on a real, operational data warehouse.

## VI. RELATED WORK

The industry analyst Gartner is a well-known authority in the business intelligence industry, and the Magic Quadrant analysis is by many considered the most influential description of the top international vendors. The Magic Quadrant analysis for 2010 [22] categorizes 15 vendors as either "market leaders" (7 companies), "challengers" (3 companies), or "niche players" (5 companies). The companies identified as "market leaders" are: IBM, Oracle, Microsoft, SAS, SAP, Information Builders, and MicroStrategy. Gartner categorizes features such as sentinels as "predictive modeling and data mining", and all "market leaders" have features within this category. However, only SAS seems to be significantly differentiated from the other "market leaders" in this category with a more comprehensive predictive offering since the company originated from forecasting and predictive modeling, whereas the other companies started as DBMS providers or as providers of reporting centric solutions. Out of all features offered by the "market leaders" [9], [10], [11], [12], [18], [24], the algorithms that to some extent resemble the functionality of sentinels are *association rules*, *sequential patterns*, and *regression techniques* (see comparison below). This also holds for the "challenger" Tibco, which is the only other noteworthy company with "predictive modeling and data mining" features. As explained in detail below, these competing techniques are distinctly different from the sentinel technology implemented by TARGIT. Moreover, TARGIT is a new entrant in the "niche player" category of the Magic Quadrant, and with reference

to Gartner's statement in Section IV, the sentinels of TARGIT are also perceived as a unique feature in the Magic Quadrant analysis. In general, TARGIT is seen by Gartner as a "niche player" with a strong foothold in the mid-market, and with a unique position in its approach to BI usability. This is very much in line with the "few clicks" approach explained in Section II, and therefore the entire concept of sentinels is unique from a market perspective.

From a scientific perspective, the idea that some actions or incidents are interlinked has been well explored in *association rules* [1]. The traditional case study of association rules has been basket-type association rules, and significant effort has been put into optimizing the original Apriori algorithm [3], [5]. In general, association rule mining seeks to find co-occurrence patterns within *absolute data values*, whereas our solution works on the *relative changes in data*. In addition, association rule mining typically works on *categorical data*, i.e., dimension values, whereas our solution works on *numerical data* such as measure values. *Sequential pattern mining* adds to the complexity of association rules by introducing a sequence in which actions or incidents take place [4], thus new optimization approaches have emerged [8], [19], [20], [23]. Sequential pattern mining allows a time period to pass between the premise and the consequent in the rule, but it remains focused on co-occurrence patterns within absolute data values for categorical data. Furthermore, our solution generates rules at the *schema level*, as opposed to the *data level*, using a contradiction elimination process. The schema-level property allows us to generate fewer, more general, rules that cannot be found with neither association rules nor sequential pattern mining. In [15] we demonstrate why sequential pattern mining does not find any meaningful rules compared to simple sentinel rule discovery, referred to as "baseline" in Section V. Compared to generalized sentinel rules, the data level nature of sequential pattern mining means that it can neither identify bi-directional rules that represent the "strongest" causal relationships, nor can it qualify such relationships with a balance assessment against a threshold. The schema level nature of generalized sentinel rules gives rise to the table of combinations (TC) and the reduced pattern growth (RPG) optimization, and such optimization can therefore not be offered by sequential pattern mining or other known optimizations for simpler "market basket"-type data such as [5]. In addition to the TC and RPG optimizations, the auto-fitting of the warning period, and the ability to combine source measures into better sentinel rules, adds to the distance from our solution to sequential patterns.

Other approaches to interpreting the behavior of data sequences are various regression [2] and correlation [7], [25] techniques which attempt to describe a functional relationship between one measure and another. In comparison, sentinel rules are a set of "*micro-predictions*" that are complementary to regression and correlation techniques. Sentinel rules are useful for discovering strong relationships between a smaller subset within a dataset, and thus they are useful for detecting warnings whenever changes (that would otherwise go unno-

ticed) in a relevant source measure occur. In [15], we provide a concrete, realistic example where nothing useful is found using correlation, while sentinel rules *do* find an important relationship within a subset of the data.

The CALM theory [13] describes how BI technologies in general, and sentinels in particular, can be integrated in an OODA loop [14]. The concept of simple sentinel rules has been described in our prior work [15], and was significantly extended into generalized sentinel rules [16], that allow multiple source measures to be combined into better rules through a quality assessment that can also auto-fit the best warning period. The user interaction in sentinel mining in the TARGIT BI Suite is described in a demo paper [17]. In comparison, this paper describes an industrial implementation of the sentinel concept from a user and data perspective. In addition, we provide detailed description of the implementation of sentinels in a standardized software package, and subject this software to several experiments on a real, operational data warehouse.

## VII. CONCLUSION AND FUTURE WORK

Motivated by the need for business users to make fast decisions, we showed how users without any knowledge of databases and data mining were able to search, schedule and receive warnings from sentinels. We described an implementation where little training is needed for a user to benefit from the sentinel technology. We demonstrated sentinel mining from both a user and a data perspective, and we specifically demonstrated how to score the best sentinels. We described in detail how an algorithm for sentinel mining was implemented in the server layer of the TARGIT BI Suite, and how the user interaction with this layer takes place. We conducted several experiments and showed that both mining and usage of sentinels is feasible with good performance for the typical users on a real, operational data warehouse. Furthermore, we identified the state of the art technologies in both industry and science, and we showed that these technologies are distinctively different from sentinels, and we pointed out where sentinels are useful when other technologies are not. In summary, we demonstrated this implementation of sentinel technology to be effective, useful, and unique.

For future work we would like to incorporate an ability to visualize sentinels was desired in order to give users an understanding of what is going on behind the scenes. In addition, the observations in Section V gave us an idea to remove the source measures that have less influence than the *SentSupp* threshold prior to running the RPG. This modification is expected to restore the effect of the RPG optimization even on data without strong sentinels. We would also like to induce more flexibility by introducing intervals to replace the fixed warning period and the offset. With regards to performance, we would like to explore a tighter integration with the underlying DBMS when testing the sentinels. It would also be interesting to conduct a qualitative study in which the predictive power of sentinel mining is compared to other techniques, e.g., sequential pattern mining, gradual rule mining, bellwether analysis, and various correlation techniques. Finally, we would like to study real-

world usage of sentinels as more customers get experience with the technology.

## ACKNOWLEDGMENT

This work was supported by TARGIT A/S, Cassiopeia Innovation and the European Regional Development Fund.

## REFERENCES

- [1] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. *In Proc. of ACM SIGMOD*, pp. 207–216, 1993.
- [2] R. Agrawal, K.I. Lin, H.S. Sawhney, and K. Shim. Fast similarity search in the presence of noise, scaling, and translation in timeseries databases. *In Proc. of VLDB*, pp. 490–501, 1995.
- [3] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. *In Proc. of VLDB*, pp. 487–499, 1994.
- [4] R. Agrawal and R. Srikant. Mining Sequential Patterns. *In Proc. of ICDE*, pp. 3–14, 1995.
- [5] S. Brin, R. Motwani, J.D. Ullman, and S. Tsur. Dynamic Itemset Counting and Implication Rules for Market Basket Data. *In Proc. of ACM SIGMOD*, pp. 255–264, 1997.
- [6] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms 2nd Ed.* MIT Press, 2001.
- [7] J. Han and M. Kamber. *Data Mining Concepts and Techniques.* (2nd Ed.) Morgan Kaufmann Publishers, 2006.
- [8] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. Hsu. FreeSpan: frequent pattern-projected sequential pattern mining. *In Proc. of KDD*, pp. 355–359, 2000.
- [9] C. Ballard, J. Rollins, J. Ramos, A. Perkins, R. Hale, A. Dorneich, E.C. Milner, and J. Chodagam. *Dynamic Warehousing: Data Mining Made Easy.* ibm.com/redbooks, September 2007.
- [10] Information Builders. *Predictive Modeling: WebFOCUS RStat.* informationbuilders.com/products/webfocus/PredictiveModeling.html, current as of June 18th, 2010.
- [11] Microsoft Corporation. *Data Mining Algorithms (Analysis Services - Data Mining).* technet.microsoft.com/en-us/library/ms175595.aspx, current as of June 18th, 2010.
- [12] MicroStrategy Corporation. *MicroStrategy Data Mining Services.* microstrategy.com/Software/Products/Service\_Modules/DataMining\_Services, current as of June 18th, 2010.
- [13] M. Middelfart. *CALM: Computer Aided Leadership & Management.* iUniverse, 2005.
- [14] M. Middelfart. Improving Business Intelligence Speed and Quality through the OODA Concept. *In Proc. of DOLAP*, pp. 97–98, 2007.
- [15] M. Middelfart and T.B. Pedersen. Discovering Sentinel Rules for Business Intelligence. *In Proc. of DEXA*, pp. 592–602, 2009.
- [16] M. Middelfart, T.B. Pedersen, and J. Krogsgaard. Efficient Discovery of Generalized Sentinel Rules. *In Proc. of DEXA*, II, pp. 32–48, 2010.
- [17] M. Middelfart and T.B. Pedersen. Using Sentinel Technology in the TARGIT BI Suite. *PVLDB 3(2)*: 1629–1632, 2010.
- [18] Oracle Corporation. *Oracle Data Mining Mining Techniques and Algorithms.* oracle.com/technology/products/bi/odm/odm\_techniques\_algorithms.html, current as of June 18th, 2010.
- [19] J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.C. Hsu. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. *In Proc. of ICDE*, pp. 215–224, 2001.
- [20] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach. *IEEE TKDE 16(11)*: 1424–1440, 2004.
- [21] N. Pendse and C. Bange. *The missing “Next Big Things”.* olapreport.com/Faileddozen.htm, current June 18th, 2010.
- [22] R.L. Sallam, B. Hostmann, J. Richardson, and A. Bitterer. *Magic Quadrant for Business Intelligence Platforms.* gartner.com/technology/media-products/reprints/oracle/article121/article121.html, current as of June 18th, 2010.
- [23] R. Srikant and R. Agrawal. Mining Sequential Patterns: Generalizations and Performance Improvements. *In Proc. of EDBT*, pp. 3–17, 1996.
- [24] SAS Corporation. *Data mining with SAS Enterprise Miner.* sas.com/technologies/analytics/datamining/miner, current as of June 18th, 2010.
- [25] Y. Zhu and D. Shasha. StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time. *In Proc. of VLDB*, pp. 358–369, 2002.