Chronicle Erkennungssysteme

Florian Bezold
Seminar Innovative Internet-Technologien und Mobilkommunikation, WS 2008/2009
Institut für Informatik, Lehrstuhl Netzarchitekturen und Netzdienste
Technische Universität München
florian.bezold@mytum.de

ABSTRACT
Das Ereigniserkennungssystem, oder Chronicle Recognition System (CRS), von dem dieser Artikel handelt, bekommt als Eingabe einen Kette von Ereignissen mit Zeitstempeln, versucht darauf auftretende vorgegebene Muster zu erkennen und generiert als Ausgabe Folge-Ereignisse oder führt Aktionen aus. Die Hauptaufgabe ist effizient komplexe Muster in Echtzeit zu erkennen.

Schlüsselwörter
Chronicle Recognition System (CRS), Situationserkennung, Ereigniserkennung.

1. Einführung – Situations Erkennung
Das Chronicle Recognition System baut grundlegend auf die Situationserkennung, die sogenannte Situation Recognition[1], auf.

Hierbei werden Situationen betrachtet, die nicht auf festen Zuständen, sondern auf Entwicklungen in sich verändernden Umgebungen basieren. Um ein solches System lauffähig zu halten, muss versucht werden, durch Quasi-Vorhersehung eine gute Interpretation davon geben können, was in einem dynamischen System passiert. Solche Aufgaben ergeben sich zum Beispiel in Umgebungs-, oder Prozessüberwachung in Netzwerken

Die Entwicklung dieser Situationserkennung orientierte sich an einer Multi-Sensor Maschine. Diese besaß mobile und feste Kameras, Laser zur Entfernungsbestimmung, Lichtschranken und Schallmesser. Sie wurde benutzt um eine geschlossene Umgebung zu überwachen.

Wenn nun also ein Sensor eine Veränderung feststellt, sei es ein Durchschreiten einer Lichtschranke oder eine Bewegungserkennung, so wird dieser Vorfall interpretiert und als Ereignis festgehalten. Eine Menge von Ereignissen, die innerhalb eines Zeitrahmens auftreten bezeichnet man als Situation. Die Feststellung einer solchen Situation, kann wiederum Ereignisse generieren oder Alarme auslösen, Nachrichten verschicken oder weitere Aktionen auslösen.

Der Entwickler eines solchen Systems stellt am Anfang Modelle von Situationen zur Verfügung, von möglichen Entwicklungen, die in der beobachteten Umgebung auftreten können. Jedes solches Situationsmodell besteht aus einer Menge von Ereignismustern und zeitlichen Beschränkungen zwischen ihnen. Wenn nun also Ereignisse beobachtet werden, die einem Muster entsprechen und auch die zeitlichen Schranken eingehalten werden, spricht man von dem Auftreten einer Instanz dieser Situation. Es kann auch angegeben werden, welche Folgeereignisse oder Aktionen beim Auftreten einer bestimmten Situation ausgelöst werden sollen. Ein solches Folgeereignis kann wiederum als Eingabe einer anderen Situation dienen, womit auch rekursive Ketten erzeugt werden können.

Die Hauptaufgabe eines solchen Systems ist es komplexe, temporale Muster zeitnah zu Erkennen, während diese auftreten.

2. Chronicle Erkennung
Die Chronicles im CRS beschreiben also solche Situationsmodelle, mit ihren Mustern von Ereignissen und deren zeitlichen Beschränkungen.

Das CRS bekommt als Eingabe eine Kette von Ereignissen, die mit Zeitstempeln versehen sind. Es erkennt Instanzen von auftretenden Chronicles, während diese sich entwickeln und produziert als Ausgabe Folgeereignisse, oder löst Aktionen aus.

Es ist grundsätzliche auf eine schnelle und effiziente Erkennung von komplexen Mustern ausgelegt.

2.1 Darstellung im CRS

2.1.1 Zeitdarstellung
Die Zeitdarstellung ist der Einfachheit halber linear gewählt, mit genügend kleinen Einheiten, um jedes Auftreten eines Ereignisses einem bestimmten Zeitpunkt zuordnen zu können.

Intervalle und Relationen können genauso benutzt werden, wie zeitliche Zusammenhänge der Zeitpunkt-Algebra (z.B. b*efore, simultaneous, after*)

2.1.2 Domänenattribute
Die Umgebung, die benutzt wird, ist durch Domänenattribute beschrieben. Ein solches Attribut besteht aus einem Paar P:v, wobei P der Attributname und v sein Wert ist.

2.1.3 Aussagen, Ereignisse
Eine Menge von Domänen Attributes P:v ist zeitlich bedingt durch Prädikate wie z.B. *event* und *hold.* Zu jedem möglichen Zeitpunkt t, hat jedes Domänenattribut nur einen einzigen Wert aus seiner Wertemenge (Abbildung 1).

Aussagen beschreiben das Bestehen des Wertes eines Attributs P über ein Zeitintervall [t1,t2], ohne exaktes Wissen darüber, wann dieser Wert erreicht wurde.

$$hold(P{:}v, (t1,t2))$$

Als Ereignismuster wird die Veränderung des Wertes eines Attributs bezeichnet. Das Ereignis selbst ist eine Instanz eines Musters, mit einem Zeitstempel, ohne bestimmte Dauer. Es wird ausgedrückt durch das Prädikat *event*.

$$event(P{:}(v_1,v_2), t)$$

Zusätzlich wird ein *forbidden event* (P, (t,t')) definiert. Das bedeutet, das Chronicle wird nicht erkannt, falls eine Veränderung des Wertes v innerhalb von t und t' auftritt.

noevent(P, (t,t'))

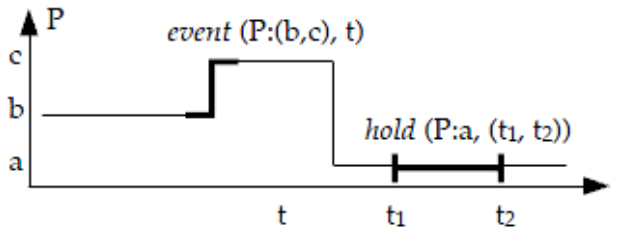

Abbildung 1: Die event *und* hold *Prädikate*

Quelle [3]

Diese drei Prädikate bilden die Basis der eigens entwickelten Sprache, zur Beschreibung der Chronicles, zu finden unter [8].

2.1.4 Ereignisverarbeitungsverzögerungen

Durch Sensorverarbeitung oder Datenübertragung wird ein Ereignis e immer erst mit einer gewissen Verzögerung empfangen. Diese Verzögerung wird begrenzt durch ein Intervall $\Delta(e)$, welches durch den Benutzer vorgegeben wird. Wenn d(e) der Zeitpunkt des Auftretens ist und r(e) der Zeitpunkt des Eingangs, so hat man immer $r(e) \in \Delta(e)+d(e)$

Diese Verzögerungen erlauben dem System Eingänge zu verarbeiten, die nicht in chronologischer Reihenfolge ankommen.

Zur Vereinfachung bezeichnet e den Namen des Ereignisses, sowie den Zeitpunkt des Auftretens d(e).

2.1.5 Chroniclemodell

Ein Chroniclemodell beschreibt einen Teil der Entwicklung der Umgebung. Es Besteht aus 4 Teilen:

- eine Menge von Ereignissen, die die relevanten Veränderungen der Umgebung für das jeweilige Chronicle repräsentieren,
- eine Menge von Aussagen, welche das Auftreten der Ereignisse des Chronicles beschreiben,
- eine Menge von zeitlichen Grenzen, und
- eine Menge von Aktionen, die durchgeführt werden, sollte das Chronicle erkannt werden.

Als Beispiel betrachten wir ein Chronicle, welches ein Problem mit einem Switch beschreibt (Abbildung 2).

```
chronicle SwitchProblem
{
    //- forthcoming chronicle events
    event (Transmission:(on,off),e1);
    event (Transmission:(off,on),e2);
    event (Component1:(?,ok),e3);
    event (Component2:(?,ok),e4);
    event (Component3:(?,ok),e5);

    //- assertions (context)
    hold (Traffic:normal, (e1,e6));

    //- temporal constraints
    e1 < e2 < e3 < e6;
    e2 < e4 < e6;
    e2 < e5 < e6;
    (e2 - e1) in [0, 180];
    (e6 - e2) in [60, 120];

    when recognised {
        report "Switch pb detection";
    }
}
```

Abbildung 2: Chronicle zur Erkennung eines Verbindungsproblems zu einem Switch

Quelle [3]

Der Block „forthcoming chronicle events“ beschreibt die Ereignisse, die der Reihe nach erwartet werden: Die Verbindung zu dem beobachteten Switch wird zum Zeitpunkt e1 unterbrochen. Nach automatischem Neustart des Gerätes, wird die Verbindung zum Zeitpunkt e2 wiederhergestellt. Danach senden die 3 Geräte, die an dem Switch angeschlossen sind, eine Statusnachricht zu den Zeitpunkten e3, e4 und e5. Das Beispiel verwendet eine einzige Aussage („assertions (context)“). Diese besagt, dass zwischen Anfang (e1) und Ende (e6) der überwachten Zeit, kein messbarer Anstieg des Datenverkehrs zu verzeichnen ist, da sonst die Reihenfolge der eingehenden Ereignisse beeinflusst werden könnte. Im Block „temporal constraints“ werden die zeitlichen Zusammenhänge definiert. Der letzte Teil des Chonicles beschreibt die Aktion, die ausgeführt wird bei vollständiger Erkennung. Fragezeichen werden an Stellen verwendet, an denen der genaue Wert nicht von Interesse ist.

2.2 Chronicle Entwicklung

Das Cronicle Recognition System muss im laufenden Betrieb alle eingehenden Daten auf Übereinstimmung mit Chronicle Modellen überprüfen. Wenn nur ein Teil der Ereignisse mit einer Untermenge von Chronicleereignissen übereinstimmen, spricht man von einer partiellen Instanz des Chroniclemodells. Bei einer kompletten Übereinstimmung (alle Zeitschranken und Aussagen werden eingehalten) spricht man davon, dass die Instanz erkannt wird.

Das System kann nur Ereignisse als Eingabe bekommen und verarbeiten. Da ein kontinuierlicher Strom von Ereignissen

angenommen wird, werden Aussagen durch Auftreten und Nicht-Auftreten dieser Ereignisse behandelt. Um z.B. die Aussage hold(P:v,(t1,t2) zu verarbeiten, überprüft das System, ob ein Ereignis *event*(P:(?,v),t) mit t<t1 stattgefunden hat, mit der Bedingung, dass keine Veränderung des Wertes von P vorgefallen ist in [t,t2[.

2.2.1 On Line Chronicle Management

Es gibt im Grunde 2 Arten, um eine Chronicleinstanz zu verändern. Entweder ein neues Ereignis tritt ein und wird in die Instanz integriert, oder Zeitlimit wird verletzt und des Instanz geschlossen.

Zur Verdeutlichung ein einfaches Szenario am eben genannten Beispiel. Die Verbindung zum Switch bricht zum Zeitpunkt 10' ab (e1), zum Zeitpunkt 12' (e2) ist der Neustart beendet. Wenn das System das Auftreten von Ereignis e1 feststellt, erkennt es eine mögliche Instanz des Chroniclemodells Es erwartet nun die Ereignisse e2, e3, e4, e5 in ihren jeweiligen Zeitfenstern (Abbildung 3 links). Ereignis e2 tritt innerhalb des Zeitfensters auf, also wird mit der Erkennung der Instanz normal fortgefahren. Wenn aber nun Ereignis e3, e4 oder e5 zum Zeitpunkt 14' nicht aufgetreten ist, wird das Zeitfenster verletzt und die Instanz geschlossen.

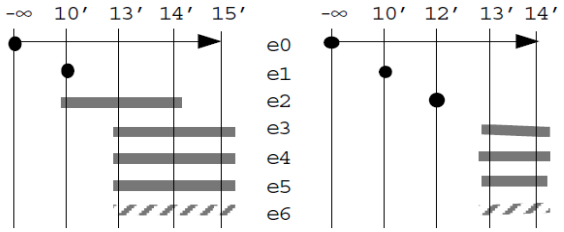

Abbildung 3: Verarbeitung von auftretenden Ereignisses innerhalb eines Chronicles

Quelle: [3]

2.2.2 Zeitlinien

Für jede Chronical Instanz werden 2 Zeitlinien geführt:

- Die *deadline* eines Chronicles ist der späteste Zeitpunkt für ein nicht auftretendes Ereignis, unter Rücksichtnahme auf evtl. anfallende Verzögerungen.
- Ähnlich berechnen wir die *non occurence line* einer Instanz, als letzter Zeitpunkt an dem man sicher sein kann, dass die letzte Aussage gilt.

Diese Zeitlinien werden benutzt, um festzustellen, ob etwas unternommen werden muss, obwohl kein Ereignis eingetreten ist. Wenn die jetzige Zeit die *deadline* überschreitet, „stirbt“ die Instanz, weil ein Folgeereignis nicht eingetreten ist. Bei Überschreitung der *non occurence line* gilt eine Aussage als veraltet (Falls es die letzte war, gilt das Chronicle als erkannt).

2.2.3 Duplikate von Chronicles

Um eine Erkennung bei jedem Auftreten zu gewährleisten, ist es von Nöten, Duplikate von Chronicles erzeugt zu erzeugen. Ohne Duplikate kann es passieren, dass Ereignisse verpasst werden, wenn sich mehrere Instanzen eines Chronicles zeitweise überlappen.

Betrachten man z.B. das folgende Modell:

e1 before e2 before e3 with (e3-e2) ≤ 3'

Als Eingabe dient die Ereigniskette:

e1 zu Zeit 0', e2 bei 3', e2 bei 8', e3 bei 11'

Das erste Auftreten von e2 wird also das Chronicle bei Zeit 6' „killen“, da kein Auftreten von Ereignis e3 innerhalb von 3' stattfindet. Das zweite Auftreten von e2 wird dadurch nicht erkannt. Diesem Problem kann entgegen gewirkt werden, indem zum Zeitpunkt 3'(erstes Auftreten von e2) eine zweite Instanz des Chronicles erzeugt wird. Eine Verdeutlichung der Erkennung von e2 zeigt Abbildung 4.

Die Hauptquelle der Komplexität des CRS ist die Anzahl der erzeugten Instanzen. Entsprechend muss versucht werden die Duplizierung zu begrenzen.

Die Erste Möglichkeit dies zu bewerkstelligen ist, jedem Chronicle eine maximale Lebensdauer zu geben. Trotz dieser Begrenzung kann immer noch eine große Anzahl von Duplikaten generiert werden.

Es kann beispielsweise vorkommen, dass Chronicles existieren, die keine zwei überlappenden Instanzen dulden. Oder der Benutzer könnte daran interessiert sein, nur eine Instanz gleichzeitig zu erkennen. In beiden Fällen wird, sobald eine Instanz erkannt wurde, alle offenen Instanzen des selben Chronicles entfernt.

2.2.4 Laufzeit

Jede elementare Operation in diesem System läuft in $O(m^2)$, wobei m die Anzahl der anstehenden Ereignisse in einer Chronicle Instanz ist. $m \leq n$ die ins gesamte Anzahl von Ereignissen im Chroniclemodell.

Für K Instanzen von Modellen mit jeweils n Ereignissen, verarbeitet der Algorithmus neue Ereignisse mit einer Komplexität von $O(Kn^2)$. K ist größer als M, die Anzahl von Chroniclemodellen. Wenn es schafft wird, dass K von der selben Ordnung wie M ist, bleibt die Gesamtkomplexität im Rahmen.

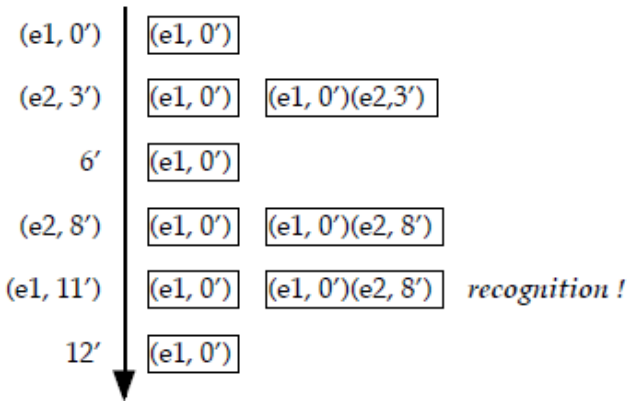

Abbildung 4: Duplikation während des Erkennungsprozesses

Quelle: [3]

2.2.5 *Frühe Anwendungen*

Zu Testzwecken wurde dieses System 1996 zur Überwachung des größten französischen Paketswitchnetzwerks eingesetzt [3]. Es wurden um die 100 verschiedene Chroniclemodelle benötigt um die selbe Arbeit zu verrichten wie das damals aktuell laufende System, welches an die 300 verschiedenen Regeln brauchte. Außerdem ist die Darstellung mit Chronicles für den Operator natürlicher, als Regeln für die Beschreibung von Abfolgen von Alarmen.

Ferner wurde es z.B. im AUSTRAL Projekt benutzt, um Alarmsequenzen zu analysieren, die von Unterstationen in einem mittleren französischen Stromverteilernetzwerk ausgesendet wurden [9].

3. Verbesserungen

Im Laufe der Zeit wurden einige Verbesserungen vorgenommen, um die Performance des Systems zu steigern. Ein paar von ihnen werden im Folgenden kurz dargestellt

3.1. Ereigniszähler

Die Hauptmotivation für diese Erweiterung entstammt der Alarmverarbeitung. Viele Alarme können von ein und demselben bestehenden Problem ausgelöst werden und das Zählen kann dabei hilfreich sein, die Gewichtung des Problems besser zu bestimmen. Außerdem konnten einige Fehler ausschließlich durch Zählen von Alarmen festgestellt werden.

So zum Beispiel im französischen Telekommunikationsnetzwerk. Die zwei folgenden Fehler sind bekannt:

- F1: ein technisches Zentrum (welches viele kleinere Unterzentren beherbergt) fährt herunter und startet neu. Es sendet der Überwachung ein „reboot“ Ereignis und jede der Unterstationen sendet ein „OK status“ Ereignis,
- F2: wenn eine Unterstation neu startet, sendet es ebenfalls ein „OK status“ Ereignis an die Überwachung.

Wenn nun viele dieser Fehler auf einmal auftreten, kann es passieren, dass einige der „reboot“ und „OK status“ Nachrichten von F1 verloren gehen und man den Ausfall von einem wichtigen Technischen Zentrum, nicht vom Ausfall von mehreren kleinen Unterstationen unterscheiden kann.

In der bisherigen Sprache sind solche Zähler schwer zu implementieren. Um diesem Problem entgegen zu wirken, wurde zusätzlich zu den bestehenden drei Prädikaten, *event*, *noevent* und *hold*, ein neues Prädikat *occurs* eingeführt:

$$occurs((n_1,n_2),a,(t1,t2), \text{ mit } 0 \leq n_1 \leq n_2$$

Dieses Prädikat beschreibt, zwischen Zeitpunkt n_1 und n_2 gibt es genau N Auftreten von Ereignissen, die dem Muster a entsprechen. Mit $n_1 \leq N \leq n_2$.

Unser neues Prädikat kann als Vereinheitlichung unser Chroniclesprache dienen, da man alle alten Prädikate durch das Neue ausdrücken kann. Das *noevent* Prädikat würde beispielsweise so aussehen:

$$noevent(a,(t1,t2)) \equiv occurs((0,0),a,(t1,t2))$$

Das Prädikat *event* sagt aus, dass ein Ereignis mindestens ein mal auftreten muss, es lautet also folgendermaßen:

$$event(a,t) \equiv occurs((1,+\infty),a,(t,t+1))$$

Diese neue Repräsentation von Zählern ist effizienter, weil es Chronicles in einer präziseren Art und Weise beschreibt. Wenn man nur mit den alten Prädikaten versucht Zähler zu implementieren, vergrößert sich die Anzahl von Chronicle Modellen, die das Erkennungssystem benutzt.

Dadurch, dass die Performance des Systems direkt mit der Anzahl der Chroniceinstanzen (oder Hypothesen) die erstellt werden zusammenhängt, vergleicht der folgende empirische Test wie viele davon mit dem alten und dem neuen System bei der Erkennung erstellt werden.

Es wurden zehn zufällige Chronicles mit Ereigniszählern, mit einer Obergrenze von weniger als 6 Ereignissen, erstellt. Außerdem wurden die entsprechenden Chronicles zusätzlich mit der alten Methode geschrieben, ohne *occurs* Prädikate (entsprach einer Anzahl von ca. 40 Chronicles). Beide Systeme bekamen als Eingabe log-Dateien, mit um die 1000 Ereignissen. Die folgende Tabelle zeigt die Anzahl der Chronicleinstanzen, die generiert wurden mit beiden Systemen.

Δ	Reco	old CRS	new CRS	ratio
4.418	165	35033	4712	7.43
7.025	105	32456	4722	6.87
9.64	71	26139	4411	5.93
12.1	58	23056	4348	5.3

Tabelle 1: Vergleich: altes und neues System

Quelle: [4]

Die Anzahl der Erkennungen (Reco) bleibt offensichtlich gleich, da sich an der Logik dahinter nichts verändert hat. Der Parameter Δ beschreibt die Verzögerung zwischen zwei aufeinander folgenden Ereignissen.

Bei einer Anwendung die z.B. das Zählen von Alarmen beinhaltet, kann mit dieser neuen Methode die Performance um ein vielfaches verbessert werden.

3.2. Temporäre Fokussierung

In manchen Fällen ist es so, dass verschiedene Ereignisse unterschiedlich oft auftreten. Angenommen, Ereignis f tritt sehr häufig auf und Folgeereignis e nur sehr selten. Wegen diesem Unterschied, könnte die Erkennung des Chronicles sehr schwierig sein. Für jedes Auftreten von f wird eine Instanz des Chronicles erstellt, die auf ein Eintreten von e wartet. Wenn also zwischen Zeitpunkt 0 und 1, tausendmal das Ereignis f auftritt, werden auch tausend Instanzen erstellt. Da aber e nur sehr selten auftritt, werden die meisten dieser Instanzen letztendlich wieder zerstört.

Um die Performance zu steigern und die Erstellung vieler unnötiger Instanzen zu verhindern, wird der Fokus auf das Ereignis e gelegt. Das geschieht folgendermaßen: wenn ein Ereignis f eintritt, wird dieses Ereignis in einem Kollektor gespeichert und eine neue Instanzen nur bei einem Auftreten von e erstellt. Wird ein Ereignis e festgestellt, prüft das System, welches vorhergehende Ereignis f im Kollektor das zu e gehörende ist, und erstellt die Instanz des Chronicles. Damit senkt man die effektive Anzahl der Instanzen die zur Erkennung von Mustern erstellt werden müssen.

Um nicht auf unwahrscheinliche Ereignisse beschränkt zu sein, wird ein *Level* für jeden Typ von Ereignissen eingeführt. Die temporäre Fokussierung sieht also so aus:

Fange mit einer Integration von Ereignis mit Level n+1 *nur an, wenn eine Instanz existiert, so dass alle Ereignisse mit Level zwischen* 1 *und* n *bereits integriert wurden.*

4. Anwendungen

Mit chroniclebasierenden Ansätzen wurde bisher in vielen Domänen experimentiert.

Das TIGER Projekt (Aguilar *et al*., 1994 [6]) entwickelte ein Softwaresystem, das die selbe durchgehende Überwachung einer Gasturbine gewährleisten sollte, die auch ein gut ausgebildeter Ingenieur bieten würde. Es bestand aus zwei Anwendungen. Einerseits wurde die 28MW General Electric Gasturbine des Fife Ethylene Kraftwerks in Schottland überwacht. Diese Turbine war ein vitaler Bestandteil des Kraftwerks, falls sie ausfiel hätte das komplette Kraftwerk abgeschaltet werden müssen. Andererseits kam das System in einigen Dassault Aviation Flugzeugen zum Einsatz und überwachte dort die „auxiliary power unit", eine kleinere Turbine. Das vom TIGER Projekt zur Verfügung gestellte System basierte teilweise auf der Chronicle Erkennung und konnte Ereignisse beschreiben, die für ein klassisches System zu komplex gewesen wären.

Weitere Anwendungen finden sich in der Medizin, z.B. zur Erkennung von Hepatitis Symptomen oder zur Intelligenten Patientenüberwachung. Das CALICOT System (Carrault et al., 1999 [7]) beschäftigt sich mit EKG Auswertung und der Erkennung von Herzrhythmusstörungen. Die On-line Analyse der EKG Daten wird von einem Chronicle Erkennungs-System durchgeführt. Es erkennt pathologische Situationen, indem es die symbolischen Beschreibungen der Signale mit zeitlichen Mustern vergleicht. In [10] werden erste Ansätze für die Anwendung von CRS für für Mobilitätsentscheidung in 3G&Beyond-Netzen beschrieben. Attribute für die Ereignisse sind in dem Fall beispielsweise Signalstärke und Paketverlustrate bei den aktuell verfügbaren Links (z.B. WLAN und UMTS).

5. Fazit

Chronicle Erkennung ist ein relativ neues System zur effizienten Erkennung von vorgegebenen Mustern innerhalb einer eingehenden Kette von Ereignissen. Im Gegensatz zu zur Zeit schon eingesetzten System, z.B. zur Analyse von Komponenten im Netzwerk, setzt es nicht auf Regeln zur Handhabung von Abfolgen von empfangenen Alarmen, sondern bietet dem Benutzer die Möglichkeit, in einer eigenen Programmiersprache sogenannte Chronicles zu definieren. Diese bestehen aus Ereignissen die im überwachten System vorkommen, verknüpft mit möglichen Nachfolgeereignissen die an feste Zeitfenster gebunden sind. Der Vorteil gegenüber konventionellen Systemen ist einerseits der hohe Grad an Formalität, mit dem die Modelle der Muster, welche überwachen werden sollen, beschrieben werden können. Anderseits die Effizienz der Erkennung, die einen Einsatz bei Echtzeit-Überwachung möglich macht.

6. Literatur

[1] C. Dousson, P. Gaborit, M. Ghallab. Situation recognition: representation and algorithms. *Thirteenth International Join Conference on Artificial Intelligence* (1993), Chambéry, pp. 166-172.

[2] C. Dousson, P. Le Magait. Improvement of chronicle-based monitoring using temporal focalization and hierarchization. *In proc. of the 17th International Workshop on Principles of Diagnosis (DX), pp. 257-261. Peñaranda de Duero, Burgos, Spain, June 2006.*

[3] C. Dousson. Alarm driven supervision for telecommunication networks: II- On-line chronicle recognition. *Annals of Telecommunications n° 9/10 (tome 51). September/October 1996, pp. 501-508*

[4] C. Dousson. Extending and unifying chronicle representation with event counters. *In proc. of the 15th ECAI, F. van Harmelen (ed.), IOS Press. pp. 257-261 Lyon, France, July 2002.*

[5] M.-O. Cordier, C. Dousson. Alarm driven monitoring based on chronicles. *In proc. of the 4th Sumposium on Fault Detection Supervision and Safety for Technical Processes (SafeProcess), pp. 286-291 Budapest, Hungary, June 2000*

[6] J. Aguilar et. al., TIGER: real-time situation assessment of dynamic systems. *Intelligent Systems Engineering pp. 103-124, 1994*

[7] G. Carrault et. al., A model-based approach for learning to identify cardiac arrhythmias. *In: AIMDM'99: Artificial Intelligence on Medicine and Medical Decision Making (W. Horn et al., Ed.). Vol. 1620 of LNAI. Springer Verlag Aalborg, Denmark.*

[8] C. Dousson. Chronicle's Language *http://crs.elibel.tm.fr/docs/language/index.html*

[9] P. Laborie, J.-P. Krivine, Automatic generation of chronicles and its application to alarm processing in power distribution systems. *8th international workshop of diagnosis (DX'97). Mont Saint-Michel, France, 1997*

[10] C. Dousson, *K.* Pentikousis, Chronicle Recognition for Mobility Management Triggers. *In: IEEE Symposium on Computers and Communications (ISCC'07), Portugal, 2007.*