OBDELAVA KOMPLEKSNIH DOGODKOV PRI SPREMLJANJU PROIZVODNEGA PROCESA

Tadej Krivec, Dejan Gradišar, Miha Glavan, Gašper Mušič

Izvleček:
Med glavnimi vidiki pri optimizaciji proizvodnje sta pridobivanje in uporaba podatkov v realnem času. Le na ta način smo lahko učinkoviti pri zagotavljanju kakovosti ali pri čim hitrejšem odkrivanju napak. V ta namen se v sodobnih sestavih uporabljajo sistemi za obdelavo kompleksnih dogodkov (CEP – ang. complex event processing), ki lahko sprejmejo različne tokove podatkov in omogočajo preproste (statične) poizvedbe na širokem naboru dinamičnih podatkov. V prispevku predstavljamo arhitekturno rešitev za sprotno obdelavo kompleksnih dogodkov, ki smo jo demonstracijsko izvedli na primeru zaznavanja napak na proizvodnih napravah z uporabo metode osnovnih komponent in testnih statistik Hotelling in SPE (ang. squared prediction error). Opisan je koncept oknenja za obdelavo velikih tokov podatkov (velepodatki). Arhitekturna zasnova temelji na platformi Microsoft StreamInsight. Podatke smo zgodovinsko in realnočasno analizirali s sistemom za upravljanje podatkovnih baz Microsoft SQL Server. Prikaz podatkov je realiziran z orodjem za poslovno analitiko Power BI.

Ključne besede:
obdelava kompleksnih dogodkov, zaznavanje napak, proizvodna analitika, PCA, Hotelling, SPE, Microsoft StreamInsight

1 Uvod

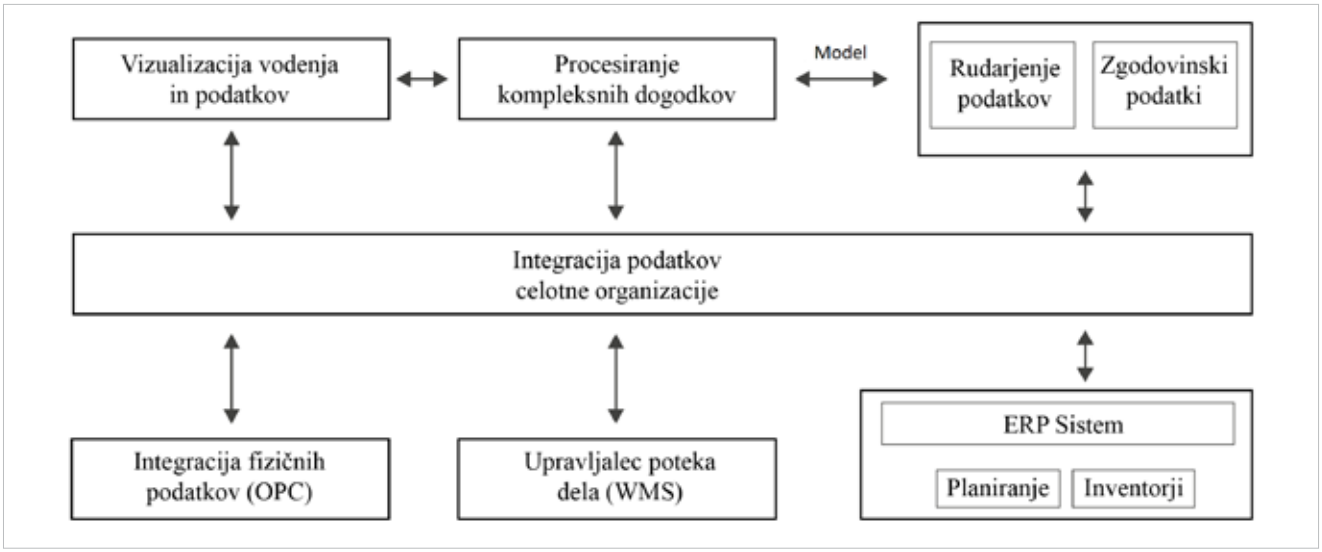
V sodobnem času hitrega razvoja tehnologije so okolja poslovnih organizacij postala zelo kompleksna. Podjetja si prizadevajo za čim večjo kakovost proizvodov, hkrati pa morajo biti stroški nizki, čas proizvodnje pa krajši. Ker se cene raznovrstnim senzorjem zmanjšujejo, je podatkov čedalje več. Ti se shranjujejo v obliki tabel (npr. Excel), baz podatkov ali pa tokove podatkov obdelujemo kar sprotno. Podatki so lahko shranjeni lokalno, z razvojem industrije in IoT (ang. internet of things) pa tudi v oblaku. Podatki se zbirajo na vseh nivojih podjetja. To veliko količino (velepodatki) in raznovrstnost podatkov je potrebno primerno agregirati in analizirati s čim manjšimi časovnimi zakasnitvami. To zahteva reorganizacijo podjetij iz obstoječih struktur v realnočasne.

Takšna obdelava podatkov je širše umeščena v koncept pametnih tovarn. Temu primerno je potrebno izbrati primerno platformo, ki podpira izvedbo takega koncepta. V nadaljevanju je predstavljena rešitev za obdelavo kompleksnih dogodkov, realizirana s platformo Microsoft StreamInsight. Rešitev je bila uporabljena na primeru zaznavanja napak (ang. fault detection) na napravah v proizvodnji in spremljanja kazalnikov učinkovitosti proizvodnega procesa. Študija primera je bila izvedena na simulacijskem modelu kemičnega procesa Tennessee Eastman [1], ki smo ga za potrebe študije še nekoliko prilagodili. Originalne časovno urejene podatke smo s simulacijskega modela pošiljali z naključnim časovnim zaostankom. Tako smo dosegli asinhrone, časovno ne urejene podatke in želeno kompleksnost za prikaz delovanja rešitve.

Mag. **Tadej Krivec**, inž., dr. **Dejan Gradišar**, univ. dipl. inž., dr. **Miha Glavan**, univ. dipl. inž., vsi Institut Jožef Stefan, Ljubljana; prof. dr. **Gašper Mušič**, univ. dipl. inž., Fakulteta za elektrotehniko, Ljubljana

2 Struktura realnočasne podpore vodenju podjetij

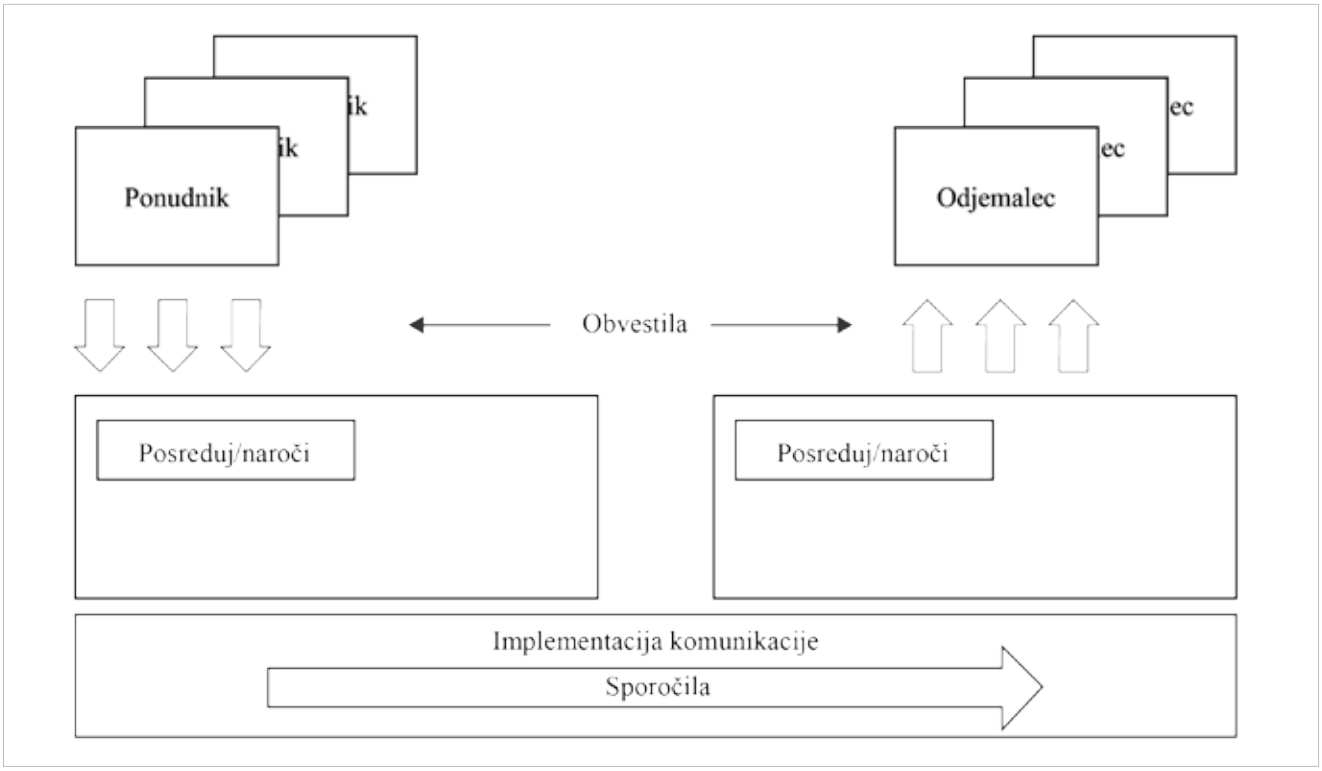
Realnočasna podpora pri vodenju podjetij je koncept upravljanja podjetij v realnem času ali blizu tega. Temu primerno je potrebno izbrati primeren informacijski sistem, kjer je pomembno, da izločimo dele odločanja, ki so implementirani z logiko pravil (ang. rule based systems) [2]. Ti so zamenjani z orodjem, ki omogoča obdelavo kompleksnih dogodkov, kjer so statična »if then« pravila realizirana v obliki modela, ki se lahko dinamično spreminja. Taka integracija podatkov je prikazana na *sliki 1*.



Slika 1 : *Integracija podatkov realnočasne podpore vodenju podjetij [3]*

2.1 Dogodkovni koncept arhitekture

V točki odločanja mora biti aktualna informacija takoj dostopna, zato je predlagan dogodkovni koncept arhitekture (ang. event driven architecture). Tak koncept je sestavljen iz ponudnika dogodkov, ki generira podatke, in odjemalca dogodkov, ki čaka na dogodke in jih ustrezno obdela. Zaznani dogodki so sprejeti v (skoraj) realnem času, tako da lahko odjemalci čim hitreje ustrezno reagirajo. Odjemalci so ločeni od ponudnikov, kar pomeni, da ponudnik ne ve, kdo od odjemalcev posluša. Tudi odjemalci so ločeni med seboj. Takšen model komunikacije se izvede po principu posreduj/naroči (ang. publish/subscribe), po katerem infrastruktura za upravljanje s sporočili sprejme vsa sporočila in zagotovi njihovo dostavo vsem možnim naročenim odjemalcem. Primer takšne komunikacije je podrobneje predstavljen na *sliki 2*. Ko je dogodek objavljen, ga sporočilni sistem posreduje vsem odjemalcem, pri tem pa pošiljatelj (ponudnik) niti ne potrebuje informacije o končnih prejemnikih. Dogodek je poslan samo



Slika 2 : *Dogodkovni koncept arhitekture [2]*

enkrat in odjemalec lahko sprejema le dogodke, ki so bili poslani po prijavi na ponudnika dogodkov. Predstavljeni koncept programiranja se uporablja, ko morajo številni podsistemi obdelati iste tokove podatkov. Primeren je za realnočasne aplikacije s čim manjšim časovnim zamikom, saj zelo hitro lahko obdela veliko količino podatkov. Primeren je tudi za računanje različnih agregacij s časovnimi okni. Prednosti so predvsem v:

- neodvisnosti ponudnikov in odjemalcev,
- enostavnem dodajanju novih odjemalcev,
- majhnih časovnih zakasnitvah,
- možnosti za enostavne razširitve,
- porazdeljeni arhitekturi,
- različnem pogledu odjemalcev na enake podatke (modularnost).

Če primerjamo storitveni in dogodkovni koncept arhitekture, so pri storitveni arhitekturi koraki pod kontrolo, tok informacije pa teče sinhrono. Eden izmed takih primerov je vertikalna integracija podatkov. Pri storitveni arhitekturi podatki potujejo z načinom poizvedbe, ta pa vrne rezultat (»ad-hoc« pristop). Pri tem so različni nivoji še vedno odvisni drug od drugega. Zato je za čisto neodvisnost primerna dogodkovna arhitektura, ki deluje z asinhronim pristopom.

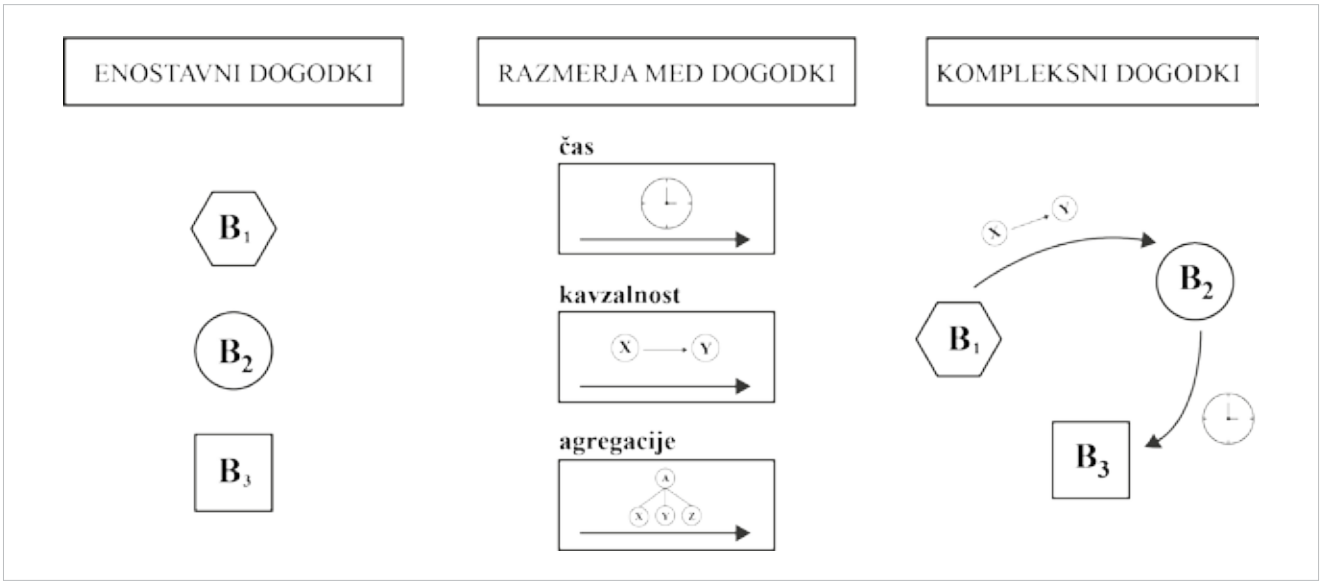
2.2 Obdelava kompleksnih dogodkov

V današnji digitalni dobi je na voljo vedno več podatkov, kar nam omogoča, da iz dogodkov razpoznamo vedno več uporabnih informacij. Množica dogodkov je lahko predstavljena z oblakom dogodkov ali tokom dogodkov. Oblak dogodkov je definiran kot delno urejena množica, v kateri je delna urejenost pogojena s časovnim, kavzalnim ali kakšnim drugim razmerjem. V praksi to pomeni, da običajno dogodki niso linearno urejeni po času. Tok dogodkov pa je, za razliko od oblaka dogodkov, linearno urejeno zaporedje dogodkov. Za različne predstavitve podatkov se uporabljajo tudi različni pristopi pri njihovi obdelavi. Obdelava dogodkovnega toka podatkov (ESP – ang. event stream processing) se uporablja pri zelo hitrih poizvedbah podatkov, ki so urejeni po času. S takim pristopom lahko prihranimo prostor v pomnilniku [2]. Po drugi strani pa lahko z obdelavo kompleksnih dogodkov (CEP) upoštevamo bolj kompleksna razmerja med dogodki, kot to na primer prikazuje *slika 3*.

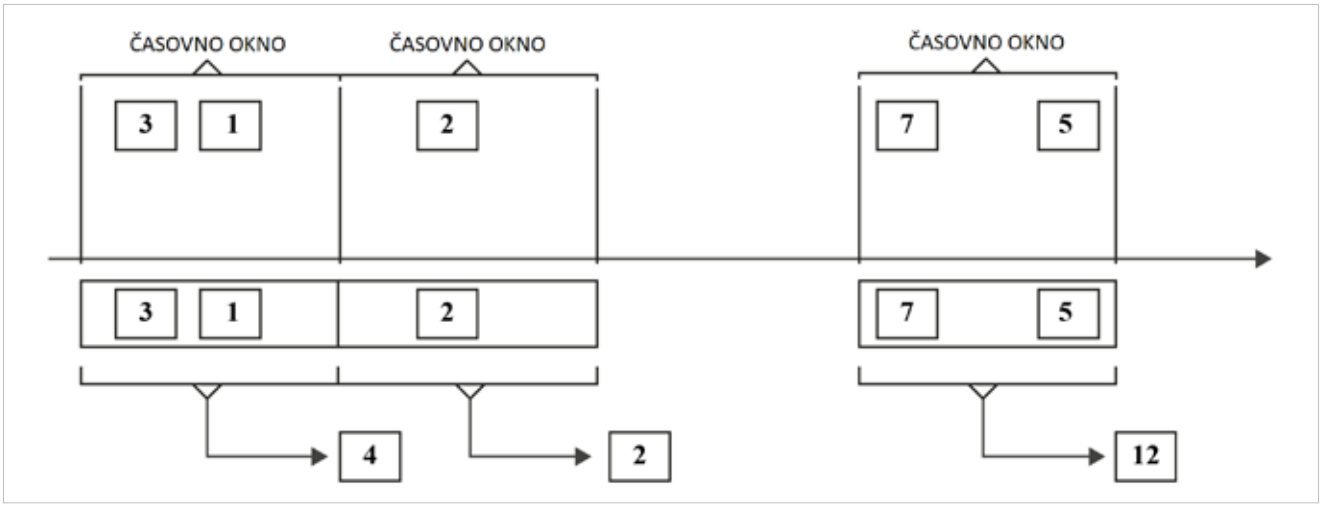
Rešitev za obdelavo kompleksnih dogodkov deluje na konceptu statičnih podatkovnih poizvedb, ki delujejo kot sito, ki filtrira in agregira podatke. Podatke pridobivamo iz virov, ki podatke pošiljajo v realnem času. Obdelani podatki se pošiljajo najprej odjemalcem, ki jih prikazujejo, shranijo ali izdajajo opozorila za uporabnike. Med postopki obdelave kompleksnih dogodkov se bolj osredotočamo na dogodke in agregacije na podatkih kot pa na podatke same [4]. Poizvedbe na podatkih so statične, podatki pa dinamični, saj konstantno prihajajo v sistem. Obdelava kompleksnih dogodkov se tako razlikuje od klasične obdelave, pri kateri so podatki statični, poizvedbe pa se izvajajo posamezno.

2.3 Oknenje

Pri obdelavi kompleksnih dogodkov se agregacije navadno računajo na množici podatkov, ki je določena s časovnim oknom. Tako lahko vsaj delno rešujemo problem velike količine podatkov. Za obdelavo kompleksnih dogodkov smo v našem primeru uporabili orodje Microsoft StreamInsight. Orodje omo-



Slika 3 : *Enostavni in kompleksni dogodki ter razmerja med njimi [2]*



Slika 4 : *Računanje vsote ob uporabi oknenja na primeru okna tipa Hopping [5]*

goča oknenje, s katerim lahko opazujemo podatke v odvisnosti od časa in v okviru katerega razpoznavamo dogodke. Za definiranje poizvedb ima Microsoft StreamInsight vgrajen namenski programski jezik LINQ (ang. Language Integrated Query).

Slika 4 prikazuje koncept računanja vsote ob uporabi oknenja. Jezik LINQ, podprt s programskim jezikom C#, na platformi Microsoft StreamInsight definira tri tipe oknenja:

- okno tipa *Count*,
- okno tipa *Snapshot*,
- okno tipa *Hopping*.

Okno tipa *Count* je definirano s številom dogodkov, ki jih vsebuje. Okno tipa *Snapshot* definira podmnožico dogodkov v časovni periodi. Okno razdeli časovnico na podlagi začetnih in končnih časov dogodkov in je zato dinamično. Okno tipa *Hopping* definira podskupine dogodkov, ki so se zgodili v določeni periodi časa. Od oken tipa Snapshot se razlikujejo po tem, da časovno os delijo na konstantne intervale, neodvisno od začetnih in končnih časov dogodkov.

3 Zaznavanje napak na industrijskih procesih

V delu smo princip obdelave kompleksnih dogodkov demonstrirali na simulacijskem industrijskem procesu, za katerega je bila razvita dogodkovno gnana platforma za izračun in prikazovanje kazalnikov proizvodne učinkovitosti (KPI – ang. key performance indicator) ter zaznavanje procesnih napak. Medtem ko so definicije KPI-jev vnaprej znane, moramo za zaznavanje napak najprej izvesti analizo arhivskih testnih podatkov, s katero določimo ustrezen model. Ta model kasneje uporabimo v okviru sprotne obdelave podatkov.

3.1 Obdelava podatkov z metodo osnovnih komponent

Metoda osnovnih komponent (PCA – ang. pincipal component analysis) je linearna transformacija, ki zmanjša dimenzijo originalnega prostora. Deluje kot nenadzorovano učenje in nam pomaga identificirati vzorce v podatkih, ki temeljijo na korelaciji med spremenljivkami. PCA išče smer največje variance in jo projicira v podprostor z enakim ali manjšim številom dimenzij. Na ta način se zmanjša kompleksnost vhodnih podatkov. Ker je metoda PCA zelo občutljiva na skaliranje podatkov, se ti najprej standardizirajo. Standardizirane vhodne podatke X označimo z Z in predstavljajo arhivske testne podatke modela Tennessee Eastman. V naslednjem koraku se izračuna kovariančna matrika Σ nad vhodnimi podatki Z. Nad kovariančno matriko nato izvedemo singularni razcep, kot to prikazuje enačba (1):

$$\Sigma = P \Lambda P^T \tag{1}$$

P predstavlja matriko lastnih vektorjev, Λ pa diagonalno matriko lastnih vrednosti matrike Z. Matriko Λ razdelimo na Λ_{PC} in Λ_{RES}, kjer Λ_{PC} vsebuje l največjih lastnih vrednosti, Λ_{RES} pa preostale. Prav tako razdelimo matriko lastnih vektorjev P na P_{PC} in P_{RES}, ki vsebujeta pripadajoče lastne vektorje lastnim vrednostim Λ_{PC} in Λ_{RES}. Spremenljivke v novem prostoru so potem definirane z enačbo (2):

$$Z' = ZP_{PC} \tag{2}$$

Metoda PCA predstavlja predhodno obdelavo podatkov, ki jo izvedemo v nesprotnem delu algoritma razpoznavanja napak. Rezultat analize je model, ki poda oceno pojava napake.

3.2 Zaznavanje napak s testno statistiko Hotelling in SPE

Testno statistiko izračunamo na podlagi primerjave vektorja povprečja učne množice in vzorca, ki pride v sistem in ga želimo prepoznati. V nesprotnem delu razpoznavanja napak izračunamo vrednosti kritičnih statistik ter povprečje ($\bar{\mu}$) in standardni odklon (σ) učne množice.

Testna statistika Hotelling je multivariantna porazdelitev, proporcionalna porazdelitvi Fisher-Snedecor (F). Je posplošitev statistike Student, ki se uporablja za testiranje multivariantnih podatkov. Kritično vrednost testne statistike Hotelling T_{KR} s stopnjo tveganja α=0,01 in prostostnimi stopnjami l in $N - l$, kjer je N število vhodnih podatkov in l število obdržanih lastnih vrednosti po obdelavi podatkov s PCA, določimo z enačbo (3):

$$T_{KR}^2 = \left(\frac{l(N-1)(N+1)}{N^2 - lN}\right) F_{\alpha,l,N-l} \quad (3)$$

Testna statistika SPE je definirana z izračunom kvadratne predikcije napake. Kritična vrednost testne statistike SPE se izračuna z enačbo (4):

$$SPE_{KR} = \Theta_1 \left(\frac{c_\alpha \sqrt{2\Theta_2 h_0^2}}{\Theta_1} + 1 + \frac{\Theta_2 h_0 (h_0 - 1)}{\Theta_1^2}\right)^{\frac{1}{h_0}}$$

$$\Theta_i = \sum_{i=l+1}^{m} \lambda_j^i, i = 1,2,3 \quad (4)$$

$$h_0 = 1 - \frac{2\Theta_1 \Theta_3}{2\Theta_2^2}$$

v kateri λ_j predstavlja lastno vrednost pri obdelavi z metodo PCA, c_α pa spremenljivko standardne porazdelitve, ki pripada zgornjemu 1-α percentilu. Rezultat nesprotnega dela algoritma so transformacijska matrika (P), diagonalna matrika lastnih vrednosti (Λ), vektor povprečja učne množice ($\bar{\mu}$), vektor standardnega odklona učne množice (σ) in kritični vrednosti testnih statistik T_{KR}^2 in SPE_{KR}.

Pri sprotnem delu algoritma izračunamo testni statistiki na podlagi živega vzorca x in , kjer je x vektor vzorca, ki ga želimo razpoznati, z pa standardiziran vektor x z vektorjem povprečja in vektorjem standardnega odklona učne množice. Vrednost testne statistike Hotelling se izračuna z enačbo (5):

$$T^2 = z^T P_{PC} \Lambda_{PC}^{-1} P_{PC}^T z \quad (5)$$

Vrednost testne statistike SPE pa z enačbo (6):

$$SPE = e^T e = (x - \bar{x})^T (x - \bar{x}) = x^T (1 - PP^T) x \quad (6)$$

Napaka na napravi se zazna v primeru, ko testni statistiki Hotelling ali SPE na živem vzorcu presegata svoji pripadajoči kritični vrednosti.

4 Primer uporabe obdelave kompleksnih dogodkov na simulacijskem primeru

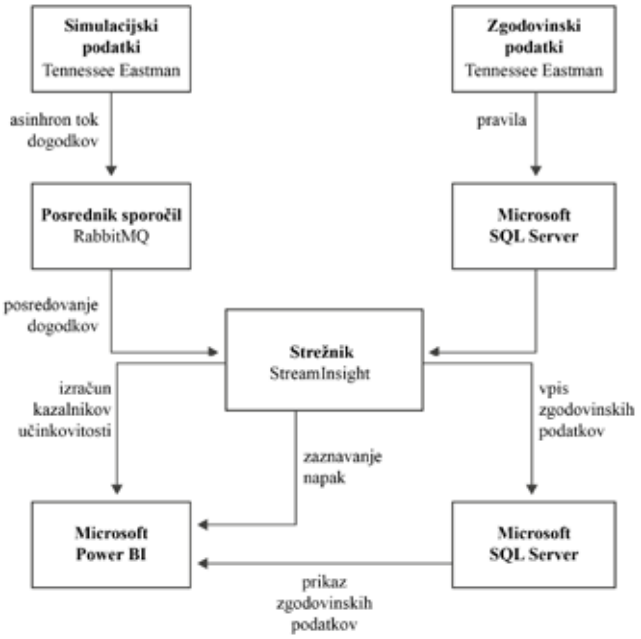
Študijo primera smo izvedli na simulacijskem modelu kemičnega procesa Tennessee Eastman [1]. Model je sestavljen v programskem jeziku Matlab & Simulink. Čeprav so bile komponente realnega procesa (kot so npr. kemijska kinetika, procesi in operacijski pogoji) modificirane zaradi zaščite podjetja, model temelji na realnem proizvodnem procesu. S simulacijskega modela (Simulink) se pošilja 41 izhodnih procesnih meritev, ki so merjene na procesu. Senzorji merijo različne dogodke in stanja na napravah, kot so dovodi v reaktor, pritisk v reaktorju, nivo reaktorja, temperatura hladilne tekočine, nivo tekočin itn. Da smo lahko prikazali učinkovitost sistema za realnočasno obdelavo kompleksnih dogodkov, smo podatke iz simulacijskega modela pošiljali z določenim naključnim časovnim zaostankom tako, da niso linearno urejeni po času. Za potrebe vodenja in analitike je na simulacijskem procesu definiran kazalnik stroškov. Ta predstavlja oceno predvidenih enournih stroškov proizvodnje glede na trenutne procesne meritve. Sestavljen je iz porabljene pare (C_1), stroška delovanja kompresorja (C_2), stroška izgub različnih komponent (C_3) in stroška izgub vsake posamezne komponente, ki zapušča proces v odvodni cevi za nečistoče (C_4). Skupni strošek obratovanja je izražen v enotah $/h in je sestavljen iz seštevka vseh delnih stroškov (C).

4.1 Arhitektura rešitve

Simulacijski model pošilja proizvodne podatke posredniku sporočil (ang. message broker). Tok podatkov s posrednika sporočil je vhod v instanco strežnika Microsoft StreamInsight, kjer se podatki obdelajo in vpišejo v bazo podatkov na Microsoft SQL Server. Microsoft StreamInsight deluje kot poslušalec toka podatkov, kar pomeni, da prejema le tok podatkov od trenutka, ko je začel poslušati na definirani izmenjavi s posrednikom sporočil. Realnočasne podatke pošilja direktno na orodje za poslovno analitiko MS PowerBI, zgodovinske podatke pa vpiše v bazo podatkov. MS PowerBI prikazuje tako realnočasne kot tudi zgodovinske podatke. Arhitektura rešitve je prikazana na *sliki 5*.

4.1.1 Posrednik sporočil

Posrednika sporočil (ang. message broker) smo implementirali z orodjem RabbitMQ, ki je odprtokodna rešitev. Ponudniki podatkov pošiljajo sporočila na izmenjave, odjemalci pa jih berejo iz podatkovnih vrst. S tem se ponudniki ločijo od vrst in se zagotovi, da ponudnikom ni potrebno skrbeti za usmerjanje podatkov. Podatkovne vrste lahko usmeri na različne strežnike in tako omogoča porazdeljen način obdelave podatkov.



Slika 5 : *Arhitektura rešitve obdelave kompleksnih dogodkov*

4.1.2 Strežnik za obdelavo kompleksnih dogodkov

Strežnik za obdelavo kompleksnih dogodkov je bil implementiran v okolju Microsoft StreamInsight. Microsoft StreamInsight je zasnovan s konceptom dogodkovne arhitekture (EDA – ang. event driven architecture) in je sestavljen modularno. *Opazovanci* so definirani s tokovi podatkov, ki vstopajo v strežnik. *Opazovanci* so moduli, ki vhodne podatke obdelajo. V študiji primera je definiran samo *opazovanec*, ki sprejme podatke s posrednika sporočil RabbitMQ. *Opazovalci* so trije in vsak predstavlja svoj modul, ki deluje neodvisno od drugega. Modul za shranjevanje zgodovinskih podatkov izračuna kazalnike učinkovitosti in jih zapiše v bazo podatkov na Microsoft SQL Server. Definiran je tudi modul za pošiljanje kazalnikov učinkovitosti na storitev Power BI.

Zaznavanje napak je realizirano v dveh delih. Nesprotni del algoritma je naučen na arhivskih testnih podatkih primera Tennessee Eastman in je realiziran v programskem jeziku Python s pomočjo knjižnice Scikit-Learn. Rezultat nesprotnega dela so parametri transformacije PCA (model zaznavanja napak), kritične vrednosti testnih statistik in vektor povprečja ter standardnega odklona učne množice. Zapisani so v podatkovno bazo. Sprotni del je realiziran kot modul na strežniku StreamInsight (v programskem jeziku C#) in bere parametre nesprotnega dela iz podatkovne baze. S pomočjo knjižnice LinearAlgebra izračuna testni statistiki in analizira, ali je prišlo do napake v proizvodnji. Rezultat zaznavanja napak je poslan na orodje API Power BI preko protokola HTTP (ang. hypertext transfer protocol). Omeniti velja, da bi bil lahko tudi sprotni del realiziran v programskem jeziku Python. Potrebovali bi le klic na spletno storitev, ki bi vrnila rezultat zaznavanja napak. To bi preprosto realizirali z definicijo novega *opazovalca*.

4.1.3 Baza podatkov

Za bazo podatkov smo uporabili platformo Microsoft SQL Server. To je sistem za upravljanje z relacijskimi bazami podatkov. Rešitev je bila zasnovana z arhitekturo zvezda, ki definira 3 dimenzijske tabele in tabelo dejstev. Ločeno je definirana tudi tabela za shranjevanje modela nesprotnega zaznavanja napak.

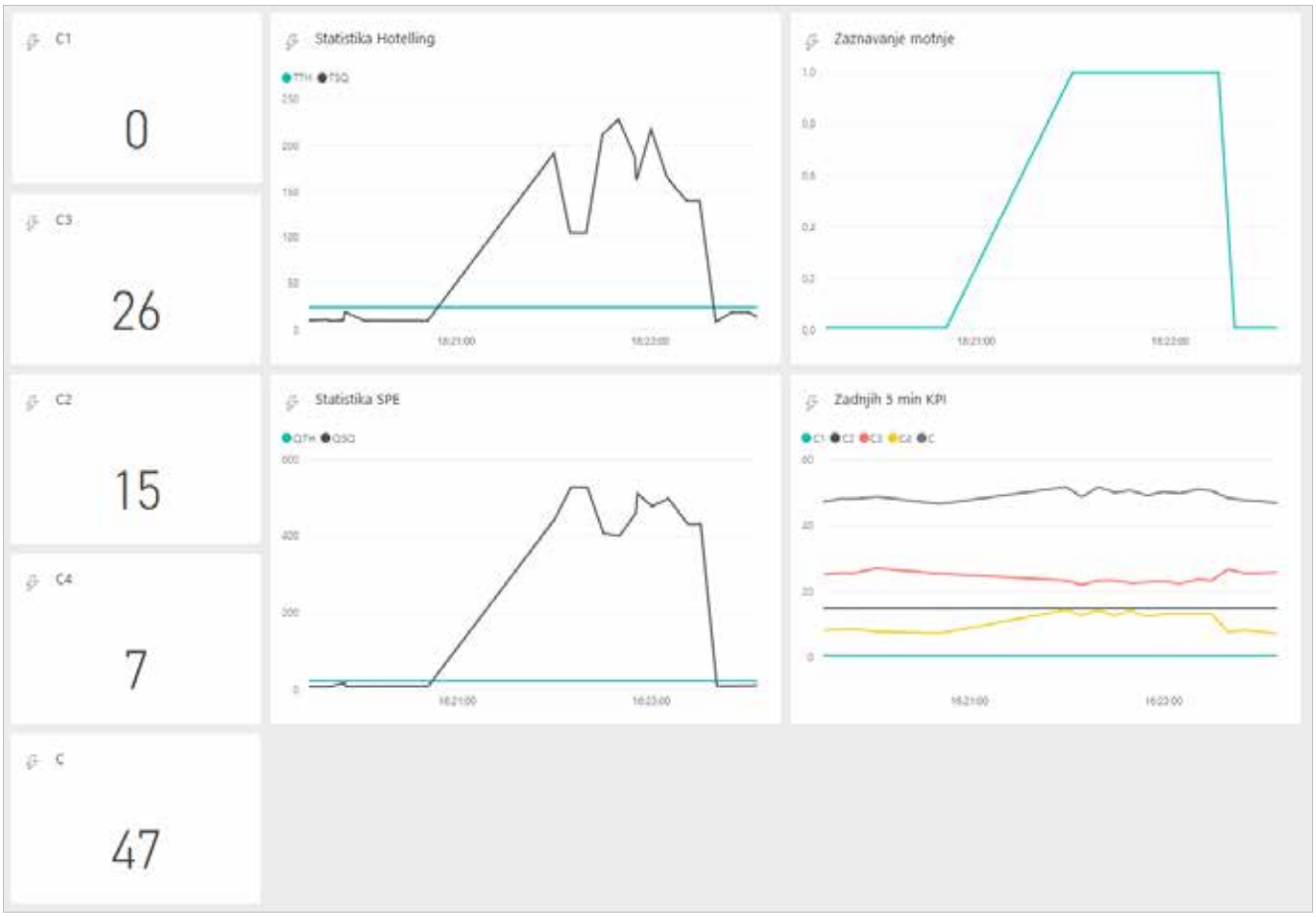
4.1.4 Nadzorna plošča spremljanja proizvodnje

Prikaz podatkov smo realizirali na storitvi Microsoft Power BI. Orodje lahko uporabljamo na dva načina: (i) kot namizno aplikacijo in (ii) kot storitev v oblaku. Mi smo nadzorno ploščo realizirali kot storitev v oblaku, kjer gradimo vizualizacije v delovnem prostoru (ang. workspace). Omogoča povezave na statične vire podatkov, kot so podatkovne baze, Excel itn. Na tok podatkov se je moč povezati na tri načine: Push dataset, Streaming dataset in PubNub streaming dataset. Na primeru smo uporabili povezavo Streaming dataset, ki omogoča prikaz podatkov v realnem času. Nadzorna plošča je sestavljena iz realnočasnega prikaza kazalnikov učinkovitosti, testnih statistik in zaznavanja napak. Prikazujejo se tudi zgodovinski podatki za zadnjih 5 minut delovanja. Nadzorna plošča je predstavljena na *sliki 6*.

Zaznavanje napak je izvedeno s sprotnim izračunom testnih statistik Hotelling in SPE. V bazi podatkov imamo shranjeni kritični vrednosti iz analize arhivskih testnih podatkov. Če testni statistiki Hotelling ali SPE presegata svoji kritični meji, se prepozna motnja. Koncept je prikazan na sliki 6 s prikazoma Statistika Hotelling in Statistika SPE, kjer črna krivulja prikazuje izračunano testno statistiko, modra črta pa prikazuje kritični meji. Na prikazu Zaznavanje motnje je prikazan rezultat zaznavanja napak, kjer vrednost 0 pomeni, da do napake ni prišlo, vrednost 1 pa, da se je zgodila napaka na napravi.

5 Zaključek

Zgradili smo rešitev, ki omogoča obdelovanje velike količine raznovrstnih podatkov in hitre obdelave podatkov z minimalnimi časovnimi zakasnitvami. Uporabili smo metodo obdelave kompleksnih dogodkov, ki omogoča enostavne statične poizvedbe na dinamičnih podatkih in združevanje raznovrstnih



Slika 6 : *Nadzorna plošča ob nastanku motnje na procesu Tennessee Eastman*

virov podatkov. Velike količine podatkov smo reševali z uporabo oknenja na toku podatkov. V pomoč nam je bil tudi posrednik sporočil RabbitMQ, ki lahko veliko količino podatkovnih tokov porazdeli po več strežnikih. Realizirano je bilo zaznavanje napak v proizvodnji na primeru Tennessee Eastman. Nesprotni del algoritma smo realizirali na arhivskih testnih podatkih v programskem jeziku Python, sprotni del pa kot modul na platformi Microsoft StreamInsight v programskem jeziku C#.

Platforma Microsoft StreamInsight se je izkazala za zelo uporabno, saj omogoča raznovrstne metode oknenja in povezovanja podatkov. Implementirati je mogoče zelo kompleksne poizvedbe, ki jih je z dobro urejeno dokumentacijo relativno lahko izvesti. Slabo lastnost predstavlja cena licence, saj potrebujemo za uporabo strežnika Microsoft StreamInsight licenco strežnika Microsoft SQL, sicer pa lahko deluje tudi kot samostojna platforma. Za lažjo integracijo med platformami smo za prikaz podatkov izbrali storitev Power BI, ki je tudi Microsoftov proizvod. Celotna rešitev je tako sestavljena s produkti podjetja Microsoft, ki omogoča enostavno povezovanje med svojimi rešitvami.

Podjetje Microsoft ponuja kot storitev tudi svojo aplikacijo Azure Stream Analytics. Deluje na oblaku Azure in je sposobna paralelnega procesiranja kompleksnih dogodkov. Omogoča tudi enostavno povezovanje s storitvijo Microsoft Power BI, kjer ima za realnočasno obdelavo definiran svoj vtičnik. Platforma Azure Stream Analytics je bolj fleksibilna in omogoča enostavno horizontalno razširitev. Prednost orodja Microsoft StreamInsight pa je, da omogoča razvoj rešitev tudi na lastni programski opremi (ang. on premise), kjer so podatki lahko zaupno shranjeni. Pri tem moramo seveda upoštevati tudi dodatne stroške nakupa strežnikov.

Podobne rešitve obdelave kompleksnih dogodkov obstajajo na platformah Apache Storm, Apache Spark, Apache Flink in Apache Kafka, ki omogočajo tako povezovanje podatkov, procesiranje velike količine podatkov in postavitev rešitve na lastni programski opremi. Omenjene platforme so odprtokodne, vendar se čas razvoja aplikacij hitro poveča. Skupaj z nakupom programske opreme tudi odprtokodne rešitve prinesejo stroške. Obstaja tudi gostovanje takih platform v oblaku, kar pa potem ne prinese bistvene stroškovne razlike od različnih oblačnih rešitev, ki jih ponujata Microsoft Azure in Amazon Web Services. Podobne rešitve ponujajo tudi ponudniki, kot so npr. SAP, Oracle, IBM, AWS in drugi [6].

Viri

[1] J. J. Downs and E. F. Vogel: A Plant-Wide Industrial Process Control Problem, Computers chem. Engng., 1993, str.: 17(3): 245–255.

[2] D. Metz: The Concept of a Real-Time Enterprise in Manufacturing, Springer Fachmedien Wiesbaden, 2014.

[3] D. Metz, S. Karadgi, and M. Grauer: A Process Model for Establishment of Knowledge-Based Online Control of Enterprise Processes in Manufacturing, International Journal on Advances in Life Sciences, 2010, 2(3 in 4):188–199.

[4] M. Meyer: Using Complex Event Processing (CEP) with Microsoft StreamInsight to Analyze Twitter Tweets 2: What are CEP and StreamInsight?, http://www.12qw.ch/2013/10/streaminsight-cep-2-what-are-cep-and-streaminsight/.

[5] Using event windows, https://msdn.microsoft.com/en-us/library/ee842704(v=sql.111).aspx.

[6] Paul Vincent: CEP Tooling Market Survey 2016, http://www.complexevents.com/2016/05/12/cep-tooling-market-survey-2016/.

Complex event processing for supervision of manufacturing process

Abstract:

One of the main aspects in optimization of production is the gathering and analysis of data in real time. This is the only way to satisfy the ever growing requirements of quality of the product or to detect faults as soon as they happen. For this purpose, complex event processing was presented in modern information systems. It allows for processing of various input data streams in enables static and simple queries on dynamic set of data. A use case is presented in fault detection on production machines with the use of PCA (principal component analysis) and test statistics Hotelling and SPE (squared prediction error). Windowing is presented as a solution to the problem of enormous amounts of data (big data). Architecture of the solution is based on the platform StreamInsight. Data is analyzed historically and in real time with the database management system Microsoft SQL Server and business intelligence tool Power BI.

Keywords:

Complex event processing, fault detection, production analytics, PCA, Hotelling, SPE, Microsoft StreamInsight

Zahvala

Delo je bilo izvedeno v sklopu programa GOSTOP, ki ga delno financirata Republika Slovenija – Ministrstvo za izobraževanje, znanost in šport – ter Evropska unija – Evropski sklad za regionalni razvoj in v sklopu nacionalnega raziskovalnega programa Sistemi in vodenje, P2-0001.