Metodi Formali per la Sicurezza

a.a. 2021/2022

Prof.ssa Francesca A. Lisi

Domenico Andrea Fabrizio Palmisano, Francesca Pollicelli

Verifica delle policy RBAC e SoD tramite Answer Set Programming

Policy, implementazione e testing

Sommario

Policy

Scenario

Lo scenario in esame riguarda lo sviluppo di un software per un committente da parte di una software house.

La software house ha diversi dipendenti e ognuno ricopre uno o più ruoli all'interno dell'organizzazione. Il flusso di lavoro per lo sviluppo del software si compone di vari task (Fig.2), ognuno assegnato ad uno o più persone aventi il ruolo qualificato per svolgerlo. Il flusso di lavoro è governato da due policy: la Role Based Access Control (RBAC) e la Segregation of Duties (SoD), al fine di proteggere l'accesso al codice sorgente dell'applicativo, eliminando i conflitti di interesse fra i ruoli dell'organizzazione.

L'obiettivo di questo progetto consiste nel modellare una soluzione in Answer Set Programming in grado di automatizzare la verifica dell'applicazione delle policy RBAC e SoD lungo tutto il flusso di lavoro.

Ruoli

- Product Owner: colui che si interfaccia con il committente del software dal primo incontro per la scelta delle specifiche dell'applicativo all'ultimo per la consegna del prodotto.
- Team Leader: colui che è a capo del team di sviluppo, organizza il lavoro del team, crea lo sprint backlog, monitora costantemente l'andamento del lavoro, valuta il lavoro fatto dai developer controllando le varie implementazioni e dando nuove istruzioni se necessario, aggiornando di conseguenza la to-do list.
- Developer: colui che si occupa direttamente dell'implementazione delle specifiche richieste, crea i vari task da eseguire, esegue le mansioni assegnate, partecipa ai daily meet per aggiornare gli altri developer sull'andamento del progetto e fornisce le varie build del software nonché la release finale.
- Question-Answer Team: team che si occupa della review delle varie build ai fini di rilevare bug creando nuovi task per la loro risoluzione e approva il passaggio alla release.
- Demo Team: team che si occupa della release per verificare se sia possibile apportare miglioramenti o è pronta alla consegna al committente.

Tasks

- Product backlog: è la fase iniziale della progettazione del software in cui il product owner esamina le specifiche richieste dal committente e definisce un elenco prioritario di funzionalità che l'applicativo dovrebbe integrare.
- Sprint backlog: esaminato il product backlog, il team leader crea tutte le attività necessarie al suo completamento e le prioritizza in una lista.
- Create tasks: vengono creati i vari task necessari a completare lo sprint.
- To do: il team leader crea una to-do list dei passi necessari per completare i task da parte dei developer.
- Implementation: i task vengono implementati da parte dei developer.
- Daily meet: riunione giornaliera a cui partecipano tutti i developer al fine di discutere circa l'andamento del lavoro, le scelte prese e se vi siano cambiamenti da attuare o vi sia qualcosa da eliminare o aggiungere alla to-do list.
- Changes: qualora sia necessario effettuare cambiamenti, vengono elencati da parte dei developer e aggiunti nella to-do list.
- Review: se non sono necessari altri cambiamenti, il team leader effettua una review sul lavoro effettuato dai developer fino a quel momento per segnalare se vi sia qualcosa che non vada bene e necessiti di essere implementato nuovamente o modificato in parte.
- New instructions: qualora il team leader notasse incorrettezze nelle implementazioni effettuate sino a quel momento, impartisce le nuove istruzioni ai developer che modificheranno o aggiungeranno qualcosa alla to-do list da implementare successivamente.
- Update to-do list: effettuata la review, il team leader aggiorna la to-do list eliminando i task correttamente implementati.
- Build: se la review del team leader va a buon fine e non vi è altro sulla to-do list, viene creata una build del software da parte dei developer.
- QA Review: il QA team effettua una review della build al fine di verificare se vi siano problemi in esecuzione o se tutto vada bene.
- Create bugs: qualora, durante la review, il QA team riscontrasse dei bug, ne viene creata una lista ed utilizzata per effettuare una nuova creazione e ripartizione dei task al fine di effettuare nuove implementazioni volte alla risoluzione dei bug riscontrati.
- Create release: se la build è priva di bug e supera la review del QA Team, viene creata la release del software ad opera dei developer.

- Meeting: il demo team effettua una review della release al fine di verificare se è possibile aggiungere migliorie sulla base delle specifiche richieste dal committente.
- Improvings: se il demo team pensa sia necessario aggiungere delle migliorie al software, ne crea una lista che viene inviata al team leader che provvederà alla creazione di un nuovo sprint backlog da cui partire per l'implementazione di tali migliorie.
- Release to customer: se la release viene dichiarata pronta e non necessita di migliorie, viene rilasciata ed inviata al committente.

RBAC e SoD

Il controllo dell'accesso basato sui ruoli (RBAC) è una policy che garantisce l'accesso alle informazioni di cui i dipendenti hanno bisogno per svolgere il proprio lavoro e impedisce loro di accedere a informazioni che non li riguardano. Il ruolo di un dipendente in un'organizzazione determina le autorizzazioni concesse all'individuo e garantisce che i dipendenti di livello inferiore non possano accedere a informazioni riservate o eseguire attività di alto livello. L'RBAC consente o meno a ciascun utente, avente ruoli e privilegi differenti, l'accesso a determinate informazioni nelle diverse modalità possibili, ossia lettura, scrittura e cancellazione.

I ruoli si basano su diversi fattori, tra cui autorizzazione, responsabilità e competenza lavorativa. Pertanto, le aziende possono designare ruoli di utente finale, amministratore o utente specializzato. Inoltre, l'accesso alle risorse può essere limitato a compiti specifici, come la possibilità di visualizzare, creare o modificare file. Nel caso specifico di questo caso di studi, i ruoli assegnati riguardano la competenza lavorativa e l'autorizzazione a leggere e/o scrivere il codice del software commissionato. Si è ipotizzato che ogni attività sia assegnata ad un solo ruolo e che ogni ruolo possa essere coinvolto in una o più attività. (Fig.1)

RUOLO	ATTIVITÀ
Product Owner	1,17
Team Leader	2,4,8,9,10
Developer	3,5,6,7,11,14,16
QA Team	12,13
Demo Team	15

Figura 1: relazione fra ruoli e attività

La Segregation of Duties (SoD) è un controllo impiegato per proteggere gli asset da accessi e/o modifiche dovute a conflitti di interesse, designando ruoli incompatibili a persone diverse. Questi contrasti insorgono nel momento in cui più ruoli condividono lo stesso oggetto: uno sviluppatore è autorizzato a leggere e modificare il codice, mentre un membro del demo team può solo leggerlo. Il conflitto scaturisce nel momento in cui un utente svolge ruoli che avrebbero dovuto essere mutualmente esclusivi: un developer non può assumere anche il ruolo di team leader in quanto quest'ultimo si occupa della revisione del codice sviluppato: se lui stesso avesse partecipato all'implementazione, il suo punto di vista non sarebbe oggettivo.

In particolare, la policy SoD utilizzata è di tipo dinamico: gli utenti possono assumere più ruoli, anche mutualmente esclusivi, purchè questi vengano assegnati in momenti diversi, ad esempio su percorsi di lavoro differenti.

Workflow

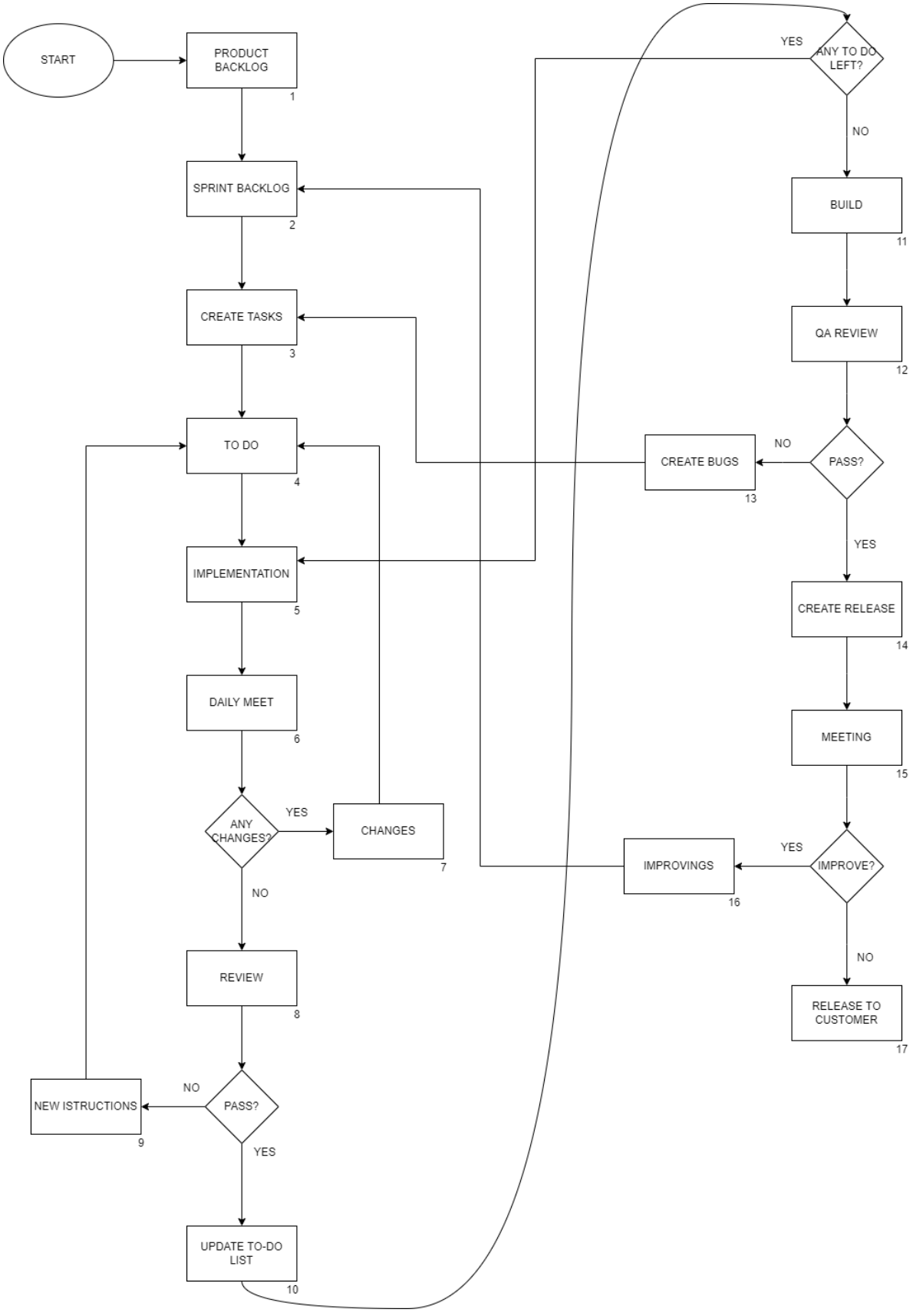

Figura 2: workflow dello sviluppo del software

Implementazione

Definiamo percorso come l'insieme di task impiegati per completare un'azione; lo scopo ultimo del progetto consiste nel verificare se tutti i percorsi del workflow possono essere completati senza violare le politiche di sicurezza RBAC e SoD, codificando il problema in Answer Set Programming.

Formalmente, ci occupiamo del problema di verifica del percorso: per un determinato path di un workflow e un insieme di persone e dei ruoli assegnati tramite la policy RBAC, ci chiediamo se sia possibile trovare un insieme di assegnazioni di ruolo per ciascuna attività nel percorso che sia conforme alle politiche SoD. Se esiste un tale insieme di assegnazioni, affermiamo che il problema è soddisfacibile.

Possiamo estendere il problema di verifica del percorso all'intero workflow, verificando che le policy siano rispettate lungo tutti i path possibili.

Segregation of Duties

Il primo passo per codificare il problema in Answer Set Programming consiste nella definizione della separazione dei compiti (Segregation of Duties, SoD) (Fig.3), impiegata per proteggere il codice da modifiche dovute a conflitti di interesse. Definiamo sod(r1,r2) la relazione di incompatibilità fra il ruolo r1 e il ruolo r2: i due ruoli devono essere separati e non possono essere eseguiti dalla stessa persona.

```
sod(team_leader,developer).
sod(qa_team,developer).
sod(demo_team,developer).
sod(qa_team,demo_team).
```

Figura 3: definizione dei SoD in ASP

Il ruolo del developer è mutualmente esclusivo rispetto al Team Leader, al QA Team e al Demo Team, mentre il QA Team è mutualmente esclusivo rispetto al Demo Team. Ciò vuol dire, ad esempio, che un developer non potrà essere nel Demo Team, poiché ha scritto il codice e conosce il suo funzionamento; quindi, il suo punto di vista in un gruppo preposto all'audit non sarebbe oggettivo.

Role Based Access Control

Dopo aver identificato i dipendenti della software house, vengono definiti i ruoli per ogni membro del personale (Fig.4).

Definiamo la relazione member $\subseteq$ P x R fra l'insieme di persone P e l'insieme dei ruoli R: member(p,r) indica che la persona p è autorizzata a ricoprire il ruolo r.

```
person(bob;alice;sofie;louis;mark;erik;mary;kelly;john;alex;tony;jenna;conny).

member(bob,product_owner).
member(louis,team_leader).
member(alice,team_leader).
member(alice,developer).
member(sofie,developer).
member(tony,developer).
member(conny,developer).
member(jenna,developer).
member(erik,developer).
member(kelly,developer).
member(mark,developer).
member(mark,qa_team).
member(alex,qa_team).
member(mary,qa_team).
member(mary,demo_team).
member(john,demo_team).
```

Figura 4: codifica di persone e membri in ASP.

Coerentemente con la definizione di SoD dinamica, alcune persone possono ricoprire più ruoli (in flussi diversi) (Fig.5).

RUOLO	PERSONA
Product Owner	Bob
Team Leader	Louis, Alice
Developer	Alice, Sofie, Tony, Conny, Jenna, Erik, Kelly, Mark
QA Team	Mark, Alex, Mary
Demo Team	Mary, John

Figura 5: specifica dei ruoli per ogni persona

Workflow model

Al fine di completare lo sviluppo del software commissionato, la software house potrebbe intraprendere diversi percorsi. Ad esempio, dopo ogni meeting e review (attività 6, 8, 12 e 15) potrebbe essere necessario tornare sui propri passi ed

eseguire nuovamente le attività precedenti. Viceversa, il flusso di lavoro potrebbe scorrere linearmente senza dover tornare indietro. L'analisi ha consentito di individuare 16 percorsi distinti.

Con pathtasks(ID,T1, T2,...,Tk) vengono definite le attività coinvolte in uno specifico percorso del workflow. La costante ID è utilizzata per identificare univocamente il percorso, mentre (T1, T2,…,Tk) sono costanti binarie che rappresentano le attività coinvolte nel percorso del flusso di lavoro. (Fig.6).

È importante evidenziare che, se un'attività viene eseguita, il numero di volte in cui questa viene ripetuta non ha alcuna influenza nella modellazione della soluzione. Per chiarire questo concetto, prendiamo in considerazione il path con ID 3: se il Team Leader non approva il codice (Review, attività 8), impartisce nuove istruzioni al team di sviluppo (New Instructions, attività 9) e gli sviluppatori aggiornano la To Do List (To Do, attività 4). Poiché l'attività 4 era già stata assegnata agli utenti con ruolo di sviluppatori, non è necessario riconsiderarla nel path.

Ragionando in questi termini, i path risultanti sono $4^2 = 16$, riassunti nella tabella (Fig.6). Le attività che precedono una scelta sono cinque (6, 8, 10, 12 e 15) ma la condizione successiva all'attività 10, in caso di risposta affermativa (Any To-Do left? Yes) non implica l'esecuzione di una nuova attività, ma torna ad un'attività già eseguita (Implementation, attività 5). Per questo motivo, scartiamo questa condizione dal numero di condizioni binarie possibili, ottenendo 4^2 path.

TASKS COINVOLTI	PATHTASK	DESCRIZIONE
1,2,3,4,5,6,8,10,11,12,14,15,17	1,1,1,1,1,1,1,0,1,0,1,1,1,0,1,1,0,1	Eseguite tutte le attività tranne 7, 9, 13 e 16
1,2,3,4,5,6,7,8,10,11,12,14,15,17	2,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,0,1	Eseguite tutte le attività tranne 9, 13 e 16
1,2,3,4,5,6,8,9,10,11,12,14,15,17	3,1,1,1,1,1,1,1,0,1,1,1,1,1,0,1,1,0,1	Eseguite tutte le attività tranne 7, 13 e 16
1,2,3,4,5,6,8,10,11,12,13,14,15,17	4,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,0,1	Eseguite tutte le attività tranne 7, 9 e 16
1,2,3,4,5,6,8,10,11,12,14,15,16,17	5,1,1,1,1,1,1,1,0,1,0,1,1,1,0,1,1,1,1	Eseguite tutte le attività tranne 7, 9 e 13
1,2,3,4,5,6,7,8,9,10,11,12,14,15,17	6,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,0,1	Eseguite tutte le attività tranne 13 e 16
1,2,3,4,5,6,7,8,10,11,12,13,14,15,17	7,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,0,1	Eseguite tutte le attività tranne 9 e 16
1,2,3,4,5,6,7,8,10,11,12,14,15,16,17	8,1,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,1,1	Eseguite tutte le attività tranne 9 e 13
1,2,3,4,5,6,8,9,10,11,12,13,14,15,17	9,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,1	Eseguite tutte le attività tranne 7 e 16
1,2,3,4,5,6,8,9,10,11,12,14,15,16,17	10,1,1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,1,1,1	Eseguite tutte le attività tranne 7 e 13
1,2,3,4,5,6,8,10,11,12,13,14,15,16,17	11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1	Eseguite tutte le attività tranne 7 e 9
1,2,3,4,5,6,8,9,10,11,12,13,14,15,16,17	12,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1	Eseguite tutte le attività tranne 7
1,2,3,4,5,6,7,8,10,11,12,13,14,15,16,17	13,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1	Eseguite tutte le attività tranne 9
1,2,3,4,5,6,7,8,9,10,11,12,14,15,16,17	14,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1	Eseguite tutte le attività tranne 13
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,17	15,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1	Eseguite tutte le attività tranne 16
1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17	16,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	Eseguite tutte le attività

Figura 6: percorsi individuati

Al fine di verificare l'intero workflow, occorre codificare tutti i path in Answer Set Programming (Fig.7). Pathid verrà utilizzato nella definizione dei test per la verifica della soddisfacibilità della soluzione generata.

```
% workflow model
pathid(1..16).
pathtasks(1,1,1,1,1,1,1,1,0,1,0,1,1,1,0,1,1,0,1).
pathtasks(2,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,0,1).
pathtasks(3,1,1,1,1,1,1,1,0,1,1,1,1,1,0,1,1,0,1).
pathtasks(4,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,0,1).
pathtasks(5,1,1,1,1,1,1,1,0,1,0,1,1,1,0,1,1,1,1).
pathtasks(6,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,0,1).
pathtasks(7,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,0,1).
pathtasks(8,1,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,1,1).
pathtasks(9,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,1).
pathtasks(10,1,1,1,1,1,1,1,0,1,1,1,1,1,0,1,1,1,1).
pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1).
pathtasks(12,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1).
pathtasks(13,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1).
pathtasks(14,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1).
pathtasks(15,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1).
pathtasks(16,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1).
```

Figura 7: codifica dei path in ASP

Mapping model

Il modello proposto da Hewett et al.[1] prevede la mappatura del problema della verifica delle policy SoD al problema noto della colorazione dei grafi. A tal fine, è necessario definire una relazione di mappatura tra il workflow descritto dal diagramma e il grafo, i cui nodi rappresentano i ruoli delle attività richieste nel flusso di lavoro, mentre gli archi rappresentano le politiche SoD fra tali ruoli: nella mappatura, i ruoli coinvolti lungo tutto il percorso devono specificare il path ID (Fig.8). I particolari ruoli associati ad ogni task vengono modellati tramite la regola role(ID,role) :- pathtasks(ID, T1,…,T17).

[1] Hewett, R., Kijsanayothin, P., Bak, S. *et al.* Cybersecurity policy verification with declarative programming. *Appl Intell* 45**,** 83–95 (2016). https://doi.org/10.1007/s10489-015-0749-8

```
role(ID,product_owner) :- pathtasks(ID,1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17).
role(ID,team_leader) :- pathtasks(ID,T1,1,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17).
role(ID,developer) :- pathtasks(ID,T1,T2,1,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17).
role(ID,team_leader) :- pathtasks(ID,T1,T2,T3,1,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17).
role(ID,developer) :- pathtasks(ID,T1,T2,T3,T4,1,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17).
role(ID,developer) :- pathtasks(ID,T1,T2,T3,T4,T5,1,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17).
role(ID,developer) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,1,T8,T9,T10,T11,T12,T13,T14,T15,T16,T17).
role(ID,team_leader) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,1,T9,T10,T11,T12,T13,T14,T15,T16,T17).
role(ID,team_leader) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,T8,1,T10,T11,T12,T13,T14,T15,T16,T17).
role(ID,team_leader) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,T8,T9,1,T11,T12,T13,T14,T15,T16,T17).
role(ID,developer) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,1,T12,T13,T14,T15,T16,T17).
role(ID,qa_team) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,1,T13,T14,T15,T16,T17).
role(ID,qa_team) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,1,T14,T15,T16,T17).
role(ID,developer) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,1,T15,T16,T17).
role(ID,demo_team) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,1,T16,T17).
role(ID,developer) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,1,T17).
role(ID,product_owner) :- pathtasks(ID,T1,T2,T3,T4,T5,T6,T7,T8,T9,T10,T11,T12,T13,T14,T15,T16,1).
```

Figura 8: mappatura ruoli-attività in ASP

Generate and Test

La generazione (della metodologia generate-and-test utilizzata in ASP) avviene attraverso l'uso di cinque regole, ciascuna delle quali assegna un ruolo ad una o più persone in un path specificato dall'ID. (Fig.9). Si è arbitrariamente scelto di assegnare le cardinalità come segue: il product owner dovrà essere solo uno, così come il team leader, i developers dovranno essere da un minimo di due ad un massimo di cinque, mentre il question-answer team e il demo team dovranno essere composti da un massimo due persone.

```
1{act(ID,product_owner,P):person(P)}1 :- role(ID,product_owner).
2{act(ID,developer,P):person(P)}5 :- role(ID,developer).
1{act(ID,team_leader,P):person(P)}1 :- role(ID,team_leader).
1{act(ID,qa_team,P):person(P)}2 :- role(ID,qa_team).
1{act(ID,demo_team,P):person(P)}2 :- role(ID,demo_team).
```

Figura 9: assegnazione dei ruoli lungo il path in ASP

Il test si compone di due vincoli (constraints to "test to eliminate" in ASP) che eliminano alcune soluzioni generate. Il primo vincolo (Fig.10) assicura che ad ogni persona, sullo stesso path, venga assegnato solo un ruolo per il quale è qualificata, rispettando le politiche RBAC definite precedentemente.

```
:- person(P),role(ID,R),pathid(ID),act(ID,R,P),not member(P,R).
```

Figura 10: constraint RBAC in ASP

Il secondo vincolo (Fig.11) rappresenta la politica di SoD: una persona, in uno stesso path, non può ricoprire le stesse cariche che entrano in conflitto per le politiche di SoD.

```
:- person(P1;P2),role(ID,R1;ID,R2),pathid(ID),act(ID,R1,P1),act(ID,R2,P2),sod(R1,R2),P1=P2.
```

Figura 11: constraint SoD in ASP

Esecuzioni

Prima esecuzione

Nella prima esecuzione viene testato l'intero workflow, comprensivo di tutti i 16 path individuati precedentemente. L'esecuzione mostra che il modello è soddisfacibile in ogni percorso; quindi, l'intero workflow è conforme alle politiche RBAC e SoD specificate.

```
clingo version 5.6.0

Reading from stdin

Solving...

Answer: 1

pathid(1) pathid(2) pathid(3) pathid(4) pathid(5) pathid(6) pathid(7) pathid(8) pathid(9) pathid(10)
pathid(11) pathid(12) pathid(13) pathid(14) pathid(15) pathid(16)
pathtasks(1,1,1,1,1,1,1,0,1,0,1,1,1,0,1,1,0,1) pathtasks(2,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,0,1)
pathtasks(3,1,1,1,1,1,1,1,0,1,1,1,1,1,0,1,1,0,1) pathtasks(4,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,0,1)
pathtasks(5,1,1,1,1,1,1,1,0,1,0,1,1,1,0,1,1,1,1) pathtasks(6,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,0,1)
pathtasks(7,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,0,1) pathtasks(8,1,1,1,1,1,1,1,1,1,0,1,1,1,1,0,1,1,1,1)
pathtasks(9,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,0,1) pathtasks(10,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,1,1,1)
pathtasks(11,1,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,1) pathtasks(12,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1)
pathtasks(13,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1,1) pathtasks(14,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,1)
pathtasks(15,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1) pathtasks(16,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)
role(1,product_owner) role(2,product_owner) role(3,product_owner) role(4,product_owner)
role(5,product_owner) role(6,product_owner) role(7,product_owner) role(8,product_owner)
role(9,product_owner) role(10,product_owner) role(11,product_owner) role(12,product_owner)
role(13,product_owner) role(14,product_owner) role(15,product_owner) role(16,product_owner)
role(1,developer) role(2,developer) role(3,developer) role(4,developer) role(5,developer) role(6,developer)
role(7,developer) role(8,developer) role(9,developer) role(10,developer) role(11,developer)
role(12,developer) role(13,developer) role(14,developer) role(15,developer) role(16,developer)
role(1,team_leader) role(2,team_leader) role(3,team_leader) role(4,team_leader) role(5,team_leader)
role(6,team_leader) role(7,team_leader) role(8,team_leader) role(9,team_leader) role(10,team_leader)
role(11,team_leader) role(12,team_leader) role(13,team_leader) role(14,team_leader) role(15,team_leader)
role(16,team_leader) role(1,qa_team) role(2,qa_team) role(3,qa_team) role(4,qa_team) role(5,qa_team)
role(6,qa_team) role(7,qa_team) role(8,qa_team) role(9,qa_team) role(10,qa_team) role(11,qa_team)
role(12,qa_team) role(13,qa_team) role(14,qa_team) role(15,qa_team) role(16,qa_team) role(1,demo_team)
role(2,demo_team) role(3,demo_team) role(4,demo_team) role(5,demo_team) role(6,demo_team) role(7,demo_team)
role(8,demo_team) role(9,demo_team) role(10,demo_team) role(11,demo_team) role(12,demo_team)
role(13,demo_team) role(14,demo_team) role(15,demo_team) role(16,demo_team) person(bob) person(alice)
person(sofie) person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex)
person(tony) person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader)
member(alice,team_leader) member(alice,developer) member(sofie,developer) member(tony,developer)
member(conny,developer) member(jenna,developer) member(erik,developer) member(kelly,developer)
member(mark,developer) member(mark,qa_team) member(alex,qa_team) member(mary,qa_team)
member(mary,demo_team) member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer)
sod(demo_team,developer) sod(qa_team,demo_team) act(1,developer,sofie) act(1,team_leader,louis)
act(1,developer,jenna) act(2,team_leader,louis) act(2,developer,kelly) act(2,developer,conny)
act(3,team_leader,louis) act(3,developer,jenna) act(3,developer,conny) act(4,team_leader,louis)
act(4,developer,erik) act(4,developer,jenna) act(5,team_leader,louis) act(5,developer,erik)
act(5,developer,kelly) act(6,team_leader,louis) act(6,developer,erik) act(6,developer,conny)
```

```
act(7,team_leader,louis) act(7,developer,erik) act(7,developer,conny) act(8,developer,sofie)
act(8,team_leader,louis) act(8,developer,kelly) act(9,team_leader,louis) act(9,developer,jenna)
act(9,developer,conny) act(10,developer,sofie) act(10,team_leader,louis) act(10,developer,erik)
act(11,team_leader,louis) act(11,developer,tony) act(11,developer,jenna) act(12,team_leader,louis)
act(12,developer,erik) act(12,developer,tony) act(13,team_leader,louis) act(13,developer,tony)
act(13,developer,jenna) act(14,team_leader,louis) act(14,developer,erik) act(14,developer,kelly)
act(15,team_leader,louis) act(15,developer,kelly) act(15,developer,tony) act(16,team_leader,louis)
act(16,developer,erik) act(16,developer,conny) act(1,qa_team,alex) act(2,qa_team,alex) act(3,qa_team,alex)
act(4,qa_team,alex) act(5,qa_team,alex) act(6,qa_team,alex) act(7,qa_team,alex) act(8,qa_team,alex)
act(9,qa_team,alex) act(10,qa_team,alex) act(11,qa_team,alex) act(12,qa_team,alex) act(13,qa_team,alex)
act(14,qa_team,alex) act(15,qa_team,alex) act(16,qa_team,alex) act(1,demo_team,john) act(2,demo_team,john)
act(3,demo_team,john) act(4,demo_team,john) act(5,demo_team,john) act(6,demo_team,john)
act(7,demo_team,john) act(8,demo_team,john) act(9,demo_team,john) act(10,demo_team,john)
act(11,demo_team,john) act(12,demo_team,john) act(13,demo_team,john) act(14,demo_team,john)
act(15,demo_team,john) act(16,demo_team,john) act(1,product_owner,bob) act(2,product_owner,bob)
act(3,product_owner,bob) act(4,product_owner,bob) act(5,product_owner,bob) act(6,product_owner,bob)
act(7,product_owner,bob) act(8,product_owner,bob) act(9,product_owner,bob) act(10,product_owner,bob)
act(11,product_owner,bob) act(12,product_owner,bob) act(13,product_owner,bob) act(14,product_owner,bob)
act(15,product_owner,bob) act(16,product_owner,bob)

SATISFIABLE


Models       : 1+

Calls        : 1

Time         : 0.285s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)

CPU Time     : 0.000s
```

Figura 12: risultati della prima esecuzione

Seconda esecuzione

Viene testato il path 8 eliminando dai ruoli un membro del demo team, due developer e un team leader. La soluzione risulta ancora soddisfacibile.

```
clingo version 5.6.0

Reading from stdin

Solving...

Answer: 1

pathid(8) pathtasks(8,1,1,1,1,1,1,1,1,0,1,1,1,0,1,1,1,1) role(8,product_owner) role(8,developer)
role(8,team_leader) role(8,qa_team) role(8,demo_team) person(bob) person(alice) person(sofie) person(louis)
person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony) person(jenna)
person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,developer)
member(sofie,developer) member(tony,developer) member(conny,developer) member(jenna,developer)
member(erik,developer) member(mark,qa_team) member(alex,qa_team) member(mary,demo_team)
sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer) sod(qa_team,demo_team)
act(8,team_leader,louis) act(8,developer,erik) act(8,developer,conny) act(8,qa_team,alex)
act(8,demo_team,mary) act(8,product_owner,bob)

SATISFIABLE


Models       : 1+

Calls        : 1

Time         : 0.056s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)

CPU Time     : 0.000s
```

Figura 13: risultati della seconda esecuzione

RUOLO	PERSONE
Product Owner	Bob
Team Leader	Louis
Developer	Erik, Conny
QA Team	Alex
Demo Team	Mary

Figura 14: riepilogo della soluzione per la seconda esecuzione

Terza esecuzione

Si testa la soluzione sul path 8, apportando alcune modifiche: assegniamo a Mark anche il ruolo di demo team e modifichiamo le cardinalità nelle regole di generazione di assegnazioni per il QA Team e il Demo Team (Fig.15).

```
2{act(ID,qa_team,P):person(P)}2 :- role(ID,qa_team).
3{act(ID,demo_team,P):person(P)}3 :- role(ID,demo_team).
```

Figura 15: generazione con nuove cardinalità

Le modifiche apportate rendono il modello non soddisfacibile.

```
clingo version 5.6.0
Reading from stdin
Solving...
UNSATISFIABLE

Models       : 0
Calls        : 1
Time         : 0.356s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

Figura 16: risultati della terza esecuzione

Quarta esecuzione

Viene inserito un nuovo product owner, aggiungendo una nuova persona member(bernadette,product_owner) ed eseguito unicamente il path 10.

```
clingo version 5.6.0
Reading from stdin
```

```
Solving...

Answer: 1

pathid(10) pathtasks(10,1,1,1,1,1,1,1,0,1,1,1,1,1,1,0,1,1,1,1) role(10,product_owner) role(10,developer)
role(10,team_leader) role(10,qa_team) role(10,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) person(bernadette) member(bob,product_owner) member(bernadette,product_owner)
member(louis,team_leader) member(alice,team_leader) member(alice,developer) member(sofie,developer)
member(tony,developer) member(conny,developer) member(jenna,developer) member(erik,developer)
member(kelly,developer) member(mark,developer) member(mark,qa_team) member(alex,qa_team)
member(mary,qa_team) member(mary,demo_team) member(john,demo_team) sod(team_leader,developer)
sod(qa_team,developer) sod(demo_team,developer) sod(qa_team,demo_team) act(10,developer,alice)
act(10,team_leader,louis) act(10,developer,erik) act(10,qa_team,alex) act(10,demo_team,john)
act(10,product_owner,bernadette)

SATISFIABLE


Models       : 1+

Calls        : 1

Time         : 0.056s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)

CPU Time     : 0.000s
```

Figura 17: risultati della quarta esecuzione

RUOLO	PERSONE
Product Owner	Bernadette
Team Leader	Louis
Developer	Alice, Erik, Alex
QA Team	Alex
Demo Team	John

Figura 18: riepilogo della soluzione per la quarta esecuzione

Quinta esecuzione

Viene testata la modalità di reasoning “brave” unicamente sul path 11, ottenendo soluzioni incrementali. Il modello risulta soddisfacibile e le soluzioni ottenute sono 11.

```
clingo version 5.6.0

Reading from stdin

Solving...

Answer: 1

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,team_leader,louis) act(11,developer,erik)
act(11,qa_team,alex) act(11,demo_team,john) act(11,product_owner,bob)
```

```
Consequences: [46;56]

Answer: 2

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,team_leader,louis) act(11,developer,erik)
act(11,qa_team,mark) act(11,qa_team,alex) act(11,demo_team,john) act(11,product_owner,bob)

Consequences: [47;56]

Answer: 3

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,team_leader,louis) act(11,developer,erik)
act(11,qa_team,mark) act(11,qa_team,mary) act(11,qa_team,alex) act(11,demo_team,john)
act(11,product_owner,bob)

Consequences: [48;56]

Answer: 4

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,team_leader,louis) act(11,developer,erik)
act(11,developer,kelly) act(11,qa_team,mark) act(11,qa_team,mary) act(11,qa_team,alex)
act(11,demo_team,john) act(11,product_owner,bob)

Consequences: [49;56]

Answer: 5

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,team_leader,louis) act(11,developer,erik)
act(11,developer,kelly) act(11,qa_team,mark) act(11,qa_team,mary) act(11,qa_team,alex)
act(11,demo_team,mary) act(11,demo_team,john) act(11,product_owner,bob)

Consequences: [50;56]

Answer: 6

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,developer,sofie) act(11,team_leader,louis)
act(11,developer,erik) act(11,developer,kelly) act(11,qa_team,mark) act(11,qa_team,mary)
act(11,qa_team,alex) act(11,demo_team,mary) act(11,demo_team,john) act(11,product_owner,bob)

Consequences: [51;56]

Answer: 7
```

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,developer,sofie) act(11,team_leader,louis)
act(11,developer,mark) act(11,developer,erik) act(11,developer,kelly) act(11,qa_team,mark)
act(11,qa_team,mary) act(11,qa_team,alex) act(11,demo_team,mary) act(11,demo_team,john)
act(11,product_owner,bob)

Consequences: [52;56]

Answer: 8

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,team_leader,alice) act(11,developer,sofie)
act(11,team_leader,louis) act(11,developer,mark) act(11,developer,erik) act(11,developer,kelly)
act(11,qa_team,mark) act(11,qa_team,mary) act(11,qa_team,alex) act(11,demo_team,mary)
act(11,demo_team,john) act(11,product_owner,bob)

Consequences: [53;56]

Answer: 9

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,team_leader,alice) act(11,developer,sofie)
act(11,team_leader,louis) act(11,developer,mark) act(11,developer,erik) act(11,developer,kelly)
act(11,developer,tony) act(11,qa_team,mark) act(11,qa_team,mary) act(11,qa_team,alex)
act(11,demo_team,mary) act(11,demo_team,john) act(11,product_owner,bob)

Consequences: [54;56]

Answer: 10

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,team_leader,alice) act(11,developer,sofie)
act(11,team_leader,louis) act(11,developer,mark) act(11,developer,erik) act(11,developer,kelly)
act(11,developer,tony) act(11,developer,conny) act(11,qa_team,mark) act(11,qa_team,mary)
act(11,qa_team,alex) act(11,demo_team,mary) act(11,demo_team,john) act(11,product_owner,bob)

Consequences: [55;56]

Answer: 11

pathid(11) pathtasks(11,1,1,1,1,1,1,1,0,1,0,1,1,1,1,1,1,1,1) role(11,product_owner) role(11,developer)
role(11,team_leader) role(11,qa_team) role(11,demo_team) person(bob) person(alice) person(sofie)
person(louis) person(mark) person(erik) person(mary) person(kelly) person(john) person(alex) person(tony)
person(jenna) person(conny) member(bob,product_owner) member(louis,team_leader) member(alice,team_leader)
member(alice,developer) member(sofie,developer) member(tony,developer) member(conny,developer)
member(jenna,developer) member(erik,developer) member(kelly,developer) member(mark,developer)
member(mark,qa_team) member(alex,qa_team) member(mary,qa_team) member(mary,demo_team)
member(john,demo_team) sod(team_leader,developer) sod(qa_team,developer) sod(demo_team,developer)
sod(qa_team,demo_team) act(11,developer,alice) act(11,team_leader,alice) act(11,developer,sofie)
act(11,team_leader,louis) act(11,developer,mark) act(11,developer,erik) act(11,developer,kelly)
act(11,developer,tony) act(11,developer,jenna) act(11,developer,conny) act(11,qa_team,mark)
act(11,qa_team,mary) act(11,qa_team,alex) act(11,demo_team,mary) act(11,demo_team,john)
act(11,product_owner,bob)

```
Consequences: [56;56]
SATISFIABLE


Models       : 11
  Brave      : yes
Consequences : 56
Calls        : 1
Time         : 0.048s (Solving: 0.00s 1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

Figura 19: risultati della quinta esecuzione

Conclusioni

La soluzione proposta e implementata in Answer Set Programming permette una efficace verifica delle politiche di Role Base Access Control e Segregation of Duties attraverso una modellazione piuttosto intuitiva di una realtà complessa come quella dello sviluppo di un software, ipotizzata in questo caso di studi: la verifica delle policy risulta flessibile ai mutamenti dell'organizzazione, non solo in termini di assegnazione di ruoli, ma anche nella gestione dei flussi di lavoro. Attraverso semplici modifiche, è possibile adattare la soluzione per la risoluzione di nuovi path, aggiungendo arbitrariamente task senza dover rimodellare l'intera soluzione.

Inoltre, questa l'implementazione, con le opportune modifiche, può essere utilizzata per modellare qualsiasi realtà che richieda la gestione di politiche di accesso; si pensi per esempio al settore finanziario, dove l'RBAC e la SoD sono indispensabili per assicurare la confidenzialità dei dati trattati e minimizzare il rischio di frodi.