Giancarlo Mangiagli

Robotic Process Automation, un approccio sostenibile

Ultimo aggiornamento: 09/12/2020

Sommario

- Premesse e scopo
- Definizioni di RPA
- RPA provvisoria
  - Definizione e contesti di applicazione
  - Utilità e usi impropri
- RPA come strumento di analisi
  - L'analisi, errori e punti deboli
  - Analisi, spunti ed opportunità
  - La RPA-riduzione
- Strumenti, framework e coordinamento
  - Classificazione e modalità d'uso
    - Macro
    - EDM
    - Framework visuali
    - Framework non visuali
  - Problemi comuni a tutti i framework
- Ottenere e gestire la sostenibilità
  - Accorgimenti suggeriti per la RPA
    - La squadra di lavoro
    - L'analisi
    - La documentazione
    - I framework
    - Sviluppo, test e produzione
    - La manutenzione
  - Un gestionale RPA
    - Architettura
    - Il database
    - L'interfaccia web
    - API e funzionalità
- Conclusioni
  - Precisazioni, riconoscimenti e ringraziamenti

Premesse e scopo

Ho scritto questo articolo dopo essermi occupato per diversi anni di Robotic Process Automation all'interno di una grande realtà aziendale. Tutte le argomentazioni non fanno riferimento a lavori precedenti o ad approcci canonici sull'argomento ma sono solo frutto di mie personali considerazioni ed esperienze sul campo. Anche i termini che ho coniato (evidenziati in grassetto lungo il testo).

La RPA è in fondo una delle possibili soluzioni ai problemi di cattiva applicazione dell'informatica. Ad esempio quando accade che due o più procedure informatiche possono comunicare tra di loro solo per il tramite di un essere umano oppure quando quest'ultimo non può ottenere da una procedura informatica elaborazioni di informazioni nei tempi e nei modi congrui alle proprie esigenze operative. Tralasciando le cause della cattiva applicazione dell'informatica, poiché lo scopo della RPA è quello di arginare questo problema, sarebbe paradossale e fallimentare che venisse applicata a sua volta in malo modo. Proprio per questa ragione, questo articolo si prefigge di:

- presentare e definire la RPA nonché un suo precipuo perimetro di azione
- enunciare alcuni dei possibili errori che si commettono con la RPA
- proporre alcuni accorgimenti che massimizzino l'efficacia d'azione della RPA

La sempre maggior diffusione di questa tecnologia, denota che negli anni si sarebbe potuto lavorare più virtuosamente nell'applicazione dell'informatica. Proprio per questa ragione ritengo che chi utilizza la RPA, che ha lo scopo di ovviare a queste carenze, la debba necessariamente applicare in modo virtuoso. In questo articolo propongo un approccio che renda la RPA sostenibile e produttiva, basato su concetti chiave come la focalizzazione sull'analisi, la provvisorietà, la rapidità di sviluppo e l'automatizzazione del controllo.

Non posso considerare come definitive le conclusioni che traggo né solidi e corretti gli argomenti che espongo. Pertanto mi riservo di operare in futuro qualsiasi opportuna revisione o aggiornamento del presente articolo.

Definizioni di RPA

I robot servono a rimpiazzare l'uomo nello svolgimento di alcune attività. La finalità di tale rimpiazzo è soprattutto quella di garantire maggiore rapidità, sicurezza e precisione rispetto al lavoro umano, e tale finalità restringe il campo di applicabilità della robotica. Banalmente, non essendo ancora tecnologicamente in grado di compiere appieno ragionamenti, i robot non possono rimpiazzare del tutto l'uomo in attività intellettive. Invece possono rimpiazzarlo in lavori, ad esempio industriali, come le catene di montaggio, in cui le sequenze di azioni sono ripetitive, ben definite e controllabili.
Controintuitivamente e paradossalmente, il recente avvento dell'era dell'informatizzazione ha portato l'essere umano a compiere nuovi tipi di azioni ripetitive e macchinose. Lo schema di queste azioni è sempre il medesimo e consta di due elementi. Il primo elemento è un programma informatico, che prende in input dei dati e restituisce in output dei risultati. Il secondo elemento è l'utilizzo del predetto programma, quindi l'atto di inserire dati in input e rilevare i dati in output. Fin dagli albori dell'informatica un programma può essere progettato per interagire con un altro programma oppure con un essere umano. Il primo caso (programma che interagisce con un altro programma) si può considerare sotto determinate condizioni una azione di Robotic Process Automation (automazione robotica di processo o processo robotico di automazione) e per questo motivo la RPA, in senso lato, esiste da quando esistono i programmi stessi (i quali sono correlati alla definizione di "processo").
Un esempio basilare di RPA è lo schedulatore, che viene programmato per innescare processi secondo determinati criteri e condizioni. Un esempio più complesso e più vicino all'odierno sentire comune sulla RPA, è un programma che interagisce con un altro programma "imitando" le azioni umane. La differenza tra i due casi è data dal fatto che il primo programma è stato predisposto dal programmatore anche per essere usato da un altro programma mentre il secondo programma è stato predisposto per essere usato solo da un essere umano.
Ci possono essere vari casi in cui un programma informatico viene pensato e predisposto nativamente per essere usato solo da un essere umano. Ad esempio:

1. Un programma di videoconferenza. Sarebbe insensato che venisse predisposto per essere usato da un altro programma.
2. Un programma di ricerca e prenotazione di voli aerei. La restrizione all'uso al solo essere umano è un tentativo di evitare che, attraverso un programma, si possano catturare grandi quantità di informazioni con alto valore commerciale.
3. Un programma di interrogazione anagrafica, interno ad una azienda. L'analista funzionale che progetta questo programma, immaginando la finalità unicamente umana di utilizzo di tale software, fa in modo che l'interazione tra esso e gli utenti sia rapida e intuitiva.

Al di là del primo caso, in cui probabilmente non vi sarà mai un interesse di automazione, negli altri due casi (specifici ma facilmente generalizzabili) invece è piuttosto semplice che maturi questo interesse.
I siti internet di ricerca comparata di voli aerei, non fanno altro che applicare la RPA su altri siti internet che forniscono il servizio finale (ad esempio quelli delle compagnie aeree) per poi rielaborare i dati raccolti e fornire le migliori offerte all'utente. Questi altri siti internet (cioè

programmi a tutti gli effetti) con cui interagiscono, come già detto, non sono nati per essere fruiti da altri programmi ma ciononostante sono prede di tentativi, spesso riusciti appieno, di cattura delle informazioni con RPA. Sono questi i casi che chiameremo **RPA strutturale**, ossia casi - nettamente in minoranza sul totale - in cui le metodologie di RPA sono tecnicamente insostituibili ed alla base stessa di un determinato software.
Il programma aziendale di interrogazione anagrafica, invece, è il tipico caso in cui, per la gestione di piccoli volumi di lavoro, l'idea di base di limitare la fruizione al solo essere umano copre per intero le esigenze aziendali e massimizza l'apporto tecnologico. Se l'addetto del comparto amministrativo necessita occasionalmente di dover interrogare una posizione anagrafica, il software dell'anagrafe deve facilitargli il compito e soddisfare appieno questa esigenza. Quando invece i volumi aziendali sono notevoli e richiedono frequenti interrogazioni, elaborazioni, riversamenti dei dati estratti su altre procedure eccetera, vengono fuori limiti e carenze del programma, che impattano pesantemente sulla produttività. Per far fronte a questo limite ci sono vari approcci, di seguito si analizzano i tre principali.
In primis un approccio ingegneristico, attuabile in fase di progettazione, basato sul seguente assunto: è impensabile per una azienda produttrice di software (non rivolto a piccole imprese), non corredare i programmi sviluppati ad uso umano da interfacce per la programmazione - quindi ad uso di altri software - che vengono definite Application Programming Interfaces (**API**). Se queste vengono progettate a monte, il costo di realizzazione è realmente minimo ma i benefici sono impareggiabilmente più alti. Pensare un prodotto anche come servizio consumabile da programmi è un indirizzo ormai diffuso capillarmente.
In secondo luogo c'è un approccio di reingegnerizzazione, valido per software già realizzati. Si basa sulla revisione del programma e sulla scelta di integrare nuove funzionalità che facciano fronte alle esigenze di lavorazioni di grandi volumi di dati direttamente da dentro il programma oppure di integrare delle API. Questa seconda opzione è vincente già nel medio periodo in quanto, via via che si dovesse far fronte a nuove esigenze, anziché rimettere mani alla modifica del programma per integrare ciò che manca (cosa che magari non viene mai fatta, data la complessità dell'intervento) è sufficiente costruire da zero con minimo sforzo piccoli programmi basati sul consumo delle API preesistenti.
Infine il terzo ed ultimo approccio consiste nell'applicazione della **RPA provvisoria**. Su quest'ultima ci si soffermerà in massima parte in questo documento, per analizzare meglio i pro, i contro, le forzature, gli effetti collaterali ed anche gli utilizzi virtuosi e sostenibili che se ne possono fare.

RPA provvisoria

Definizione e contesti di applicazione

Per RPA provvisoria si intende la realizzazione di uno o più programmi informatici i quali:

- interagiscono con altri programmi non necessariamente predisposti per tale interazione
- non sono necessariamente sviluppati in modo sinergico con i programmi con cui interagiscono (ad esempio tipicamente la squadra degli sviluppatori di un'automazione che interagisce con un sito web esterno all'azienda non coopera allo sviluppo di quest'ultimo né tantomeno viene informata su eventuali modifiche)

Queste proprietà implicano una **connotazione ragionevolmente temporanea e/o instabile della soluzione di RPA** [Di Mauro, 2018], in quanto, a meno di prevedere, catturare ed inseguire in continuazione e rischiosamente le evoluzioni dei programmi con cui interagiscono, è opportuno adottare prima possibile una soluzione più robusta e duratura.
Un altro aspetto da considerare è il contesto in cui il programma di RPA provvisoria può andare ad interagire. Si può avere un contesto **RPA-favorevole**, in cui si può ottenere pieno accesso ad esempio ad ambienti di test dei programmi da automatizzare oppure alle loro analisi funzionali, se non addirittura alle basi dati (questo avviene, ad esempio, all'interno di aziende medie e grandi in cui il comparto IT, che ha prodotto o gestito i programmi da automatizzare, mette a disposizione queste risorse). Si può avere invece un contesto **RPA-non favorevole**, come ad esempio l'interazione con un software per il quale non si ha a disposizione nessuna delle succitate risorse o peggio ancora in cui tale software presenta pochi appigli tecnici (come elementi grafici non etichettati univocamente) oppure si ha una sola credenziale di accesso e solamente in ambiente di produzione, per cui diventa complicato testare l'automazione in fase di sviluppo.

Utilità e usi impropri

La RPA provvisoria, se ben applicata, risulta utile in due macro categorie di applicazione.
La prima, come estensione del concetto di "prova di fattibilità" (Proof Of Concept), in un contesto RPA-favorevole. Tante volte si pensa che automatizzare un processo apporti notevoli benefici, in termini ad esempio di rapidità, sicurezza e quindi, in sostanza, un risparmio di impiego di forza lavoro, misurabile in FTE (full-time equivalent). Se si vuole avere la certezza che una automazione robusta e duratura sia un buon investimento, si può provare ad adottare la RPA provvisoria come strumento dimostrativo, partendo dal presupposto che si possa mettere in piedi un'automazione in molto meno tempo rispetto alla programmazione "tradizionale", a prescindere dalla qualità del risultato finale. Tale prototipo, anche qualora fosse saturo di debiti tecnici, permetterebbe di misurare nel brevissimo periodo e con rigore l'effettivo risparmio di FTE, a supporto della decisione di attivare immediatamente o meno l'implementazione di un software alla maniera tradizionale (o di API) che consolidi i comprovati benefici sulla produttività.
La seconda categoria è RPA come soluzione tampone. Il caso si dirama in più esempi:

- un'attività da svolgere una tantum, specie se con scadenze stringenti
- un'attività da svolgere imprescindibilmente e immediatamente
- un'attività in cui si interagisce con applicativi extra-aziendali che non espongono API

In questi casi è auspicabile vagliare una soluzione di RPA provvisoria che, talvolta anche nei contesti meno favorevoli, dà risultati accettabili e ha una modalità di sviluppo e rilascio rapido. Nel caso dell'attività una tantum, il prodotto RPA può essere dismesso subito dopo la

conclusione della stessa. Nel caso dell'attività imprescindibile, il prodotto RPA, rilasciato in tempi brevi, darà il tempo agli sviluppatori "tradizionali" di creare il programma definitivo che normalmente richiede tempi di sviluppo lunghi. Infine, nel caso di attività su applicativi extra-aziendali, in attesa che i produttori di questi creino apposite API o che si valuti l'opportunità di rimpiazzare il prodotto con uno dotato di API, l'interazione attraverso una soluzione RPA può tentare di dare un po' di respiro all'operatività quotidiana.
Un discorso a parte va fatto per gli usi impropri. La soluzione RPA ingolosisce facilmente gli addetti ai lavori, che cadono nel tranello di paventare risoluzioni miracolistiche e istantanee di problemi magari annosi, delicati e complessi. L'apparente semplicità con cui ci si può approcciare a prodotti commerciali di RPA, dà la sensazione di avere a portata di mano uno strumento agile, potente e rapido col rischio di trascurare aspetti cruciali quali modularità, scalabilità, sicurezza, stabilità, durabilità e affidabilità.
E' sufficiente che si verifichi uno solo di questi tre presupposti per farne uso improprio:

1. **adoperare la RPA provvisoria come soluzione definitiva**
2. **adoperare la RPA in contesti in cui la stessa è inadeguata**
3. **in un'ottica di uso massiccio di RPA, non stabilire solide metodologie di sviluppo né utilizzare strumenti di armonizzazione e controllo delle automazioni**

Le conseguenze dell'uso improprio possono essere molteplici. Innanzitutto occorre fare delle distinzioni qualitative e quantitative. Ad esempio, dal punto di vista qualitativo, un conto è affidare lo svolgimento di una attività inadeguata o rischiosa (presupposto n. 2) ad un programma RPA e un conto invece è affidarsi ad un software su misura, che rispetti i necessari requisiti. Dal punto di vista quantitativo, un conto è, fatta l'analisi rischi/benefici, decidere di affidare ad un programma RPA lo svolgimento di una sola attività inadeguata o rischiosa e un altro conto è decidere di affidare molteplici attività, anche solo mediamente inadeguate o rischiose. Difatti rilevare, correggere errori e correre ai ripari in tempi congrui rispetto alla aspettative di tempestività della RPA è un'attività onerosa, specie se riguarda un gran numero di programmi operanti ed ancor peggio in presenza del presupposto n. 3 (no metodologie / strumenti di controllo etc.). Il presupposto n. 1 è il più subdolo perché, a ben riflettere, può portare ad un disastro irrecuperabile. I programmi non RPA possono essere "indipendenti", cioè autoconsistenti, in quanto non dipendono dall'interazione con nessun altro programma. O perlomeno se interagiscono con un altro programma, lo fanno a mezzo di API delle quali, per definizione, vengono rese note in anticipo eventuali modifiche od evoluzioni, così da avere il tempo di modificare e far evolvere a sua volta il programma che le utilizza. I programmi RPA, invece, sono "dipendenti" da altri programmi ma a scatola chiusa. Per questa ragione, mentre per i programmi non RPA la frequenza di errori e blocchi è sparsamente distribuita nella linea temporale, paradossalmente può accadere per tutti o molti dei programmi RPA contemporaneamente che avvengano modifiche sui programmi da cui dipendono, senza naturalmente alcun preavviso, causando di fatto una situazione ingestibile. Se la RPA provvisoria viene utilizzata come soluzione definitiva, accade che la quantità di automatismi operanti sia sempre maggiore e, in proporzione a questa quantità crescente, lo scenario di disastro può diventare irrecuperabile. Naturalmente aumentare il numero dei programmatori RPA con l'aumento degli automatismi gestiti non è una soluzione al problema: il raddoppio dei programmatori non implica il dimezzamento del tempo di risoluzione dei problemi, in quanto non si tratta di attività "meccaniche" e "schematiche" bensì di attività dipendenti da ingegno, intuizione, competenza, fortuna ed altri fattori

difficilmente quantificabili. L'unica soluzione al problema è limitare, con criteri di buon senso e di progressivo sviluppo di "nuovo" a dismissione di "vecchio", il numero massimo di programmi RPA gestiti, mantenendo così la caratterizzazione provvisoria della RPA stessa di cui si è parlato.
A proposito infine del presupposto n. 3, è interessante coniare un termine, poco accademico e molto spregiativo, ma utile forse a tenere a mente rischi e pericoli di questo abuso di RPA: i "**manualismi**". Si tratta di una storpiatura del termine "automatismo" e subentra quando l'interazione tra un programma e un altro richiede un intervento umano, più o meno costante, più o meno oneroso. E' piuttosto frequente che con regolare cadenza un programma RPA si "blocchi" o dia errori, soprattutto se non viene sviluppato e gestito nel modo corretto ed è tipicamente altrettanto frequente adottare come soluzione, seppure poco brillante ed efficace, il continuo intervento umano per sbloccare, aggiustare, riavviare eccetera. Il manualismo può giovare a mascherare problemi strutturali e connaturati ad una cattiva applicazione della RPA ma ha come limite la qualità scadente dell'artefatto e soprattutto quantità e frequenze eccessive di problemi da risolvere. Combinando insieme il fatto che la RPA permette di abbattere l'impiego di risorse umane e nel contempo non vieta di rendere metodico un suo cattivo uso, certe situazioni possono degenerare facilmente da un momento all'altro creando vere e proprie crisi: ad esempio può accadere che gli sviluppatori RPA perdono il controllo della situazione, non riuscendo più a gestire la quantità di interventi manutentivi, e contemporaneamente non ci sono abbastanza impiegati operativi per rimpiazzare in via temporanea gli automatismi (o, meglio, i *manualismi*) con il lavoro manuale.
Seguire sempre le buone pratiche consigliate e acquisire l'esperienza di chi ha sviluppato per anni la RPA con volumi significativi, può consentire di scongiurare nella maggior parte degli scenari le circostanze negative fin qui descritte.

RPA come strumento di analisi

L'analisi, errori e punti deboli

L'approccio all'analisi dei processi da sviluppare con RPA potrebbe assumere la conformazione "provvisoria" della RPA stessa e caratterizzarsi quindi da una visione semplicistica e sbrigativa del da farsi. In pratica l'approccio più frequente - quasi sempre inopportuno - è quello di descrivere un processo come mera registrazione dei passi che compie l'operatore umano interagendo con i programmi. Segue un esempio. Occorre automatizzare un processo di estrazione di dati anagrafici da una pagina intranet. L'analista funzionale RPA intervista l'utente ed annota le varie fasi: apertura link del sito web, click sul pulsante "anagrafe", inserimento nelle caselle dei moduli del codice fiscale o della partita iva del soggetto da ricercare, click sul pulsante "ricerca", attesa e cattura del risultato, compresi eventuali messaggi di errore. Un documento del genere, che ha molto di narrazione e poco di analisi, può andare bene nei casi in cui occorre sviluppare automazioni "usa e getta", da

mettere in piedi in fretta ed usare una tantum per esigenze operative e da cui trarre quindi il maggior vantaggio possibile al netto di errori o blocchi mal gestiti. In tutti gli altri casi, rappresenta una sterile descrizione dello stato dell'arte, che non offre valore aggiunto per molti motivi. Ossia tale mera descrizione:

- può essere parziale e omissiva
- può contenere passi ridondanti
- può non essere supportata da basi normative o regolamentari
- può non tenere conto di *desiderata* evolutivi ed ottimizzazioni

L'approccio di analisi limitato all'intervista con l'utente finale porta quasi con assoluta certezza ad uno o più di questi problemi. Invece, con i dovuti accorgimenti, l'analisi finalizzata alla realizzazione di un artefatto RPA può trasformarsi in opportunità per migliorare l'intero processo lavorativo, quindi anche e non solo la parte informatica.

Analisi, spunti ed opportunità

Un approccio più costruttivo all'analisi, in linea generale, richiede alcuni, minimi, sforzi aggiuntivi che in molti casi possono offrire svariati benefici.

Innanzitutto l'approccio deve essere corale, ossia è opportuno coinvolgere in sede di analisi sia gli utilizzatori finali, sia i referenti per norme e regolamenti, sia gli analisti tecnici che quelli funzionali. Di norma gli utenti finali accettano così come sono le procedure che gli vengono fornite, o perché presumono che siano funzionalmente il massimo possibile (non essendo né analisti né tecnici) o perché non hanno alternative o particolare voce in capitolo in azienda. Ma d'altro canto le funzionalità devono essere compatibili con le loro esigenze, le loro risorse e i loro tempi di lavoro, quindi è bene coinvolgerli in maniera mirata. Gli analisti, di contro, fanno bene il loro mestiere ma tendono ad essere autoreferenziali, a meno che non siano costretti a ragionare nelle fasi cruciali in presenza degli utenti finali.

In secondo luogo, l'analisi deve andare a fondo, scavando fino all'origine dell'esigenza per confermare la validità attuale della stessa, non solo nella sostanza ma anche nella modalità di risoluzione del problema.

In terzo luogo, occorre scoprire se non ci sono modi alternativi, più immediati e sicuri per ottenere lo stesso risultato, non necessariamente con RPA. Quest'ultimo punto è quello che può aprire a opportunità fino a quel momento inesplorate: la regola è che se non prima si pone un problema, formalmente il problema non esiste ma una volta formalizzato diventa l'occasione per scoprire che potrebbe essere meno complicato di quanto si pensi se non addirittura un non-problema.

Per meglio concretizzare queste considerazioni, si può fare un esempio basato sulla lettura dell'anagrafe da intranet citata nel precedente paragrafo. Realizzando il programma di RPA sulla base del semplice documento descrittivo (già vivamente sconsigliato), si potrebbe non scoprire mai la finalità di quell'azione. Ad esempio lo scopo poteva essere quello di accumulare una serie di indirizzi postali da copiare ed incollare su modelli di lettere preconfezionate, provenienti da un altro programma. In uno scenario del genere, una migliore analisi avrebbe ottenuto risultati decisamente diversi attraverso i seguenti passi:

- Riunione di tutti i soggetti da coinvolgere (utente, referente normativo, analista funzionale e tecnico)

- Accertamento, con referenti normativi e utente finale, dello scopo e della necessità di tale procedura
- Revisione critica, da parte degli analisti, del racconto di referenti normativi e utenti finali, a proposito sia dell'esaustività delle casistiche coperte, sia di possibili eccezioni o errori, sia di futuri ed imminenti sviluppi normativi e procedurali che possano mettere in discussione l'intervento stesso eccetera. Più domande dubitative si pongono, meglio si prevengono successivi inconvenienti
- Effettuazione di simulazioni reali che coprano tutte le possibili casistiche (che offrono spunti per sollevare ulteriori quesiti costruttivi)
- Scrittura e condivisione con i soggetti interessati di un resoconto dell'analisi condotta

Ritornando all'esempio dell'anagrafe, attraverso questi passi si potrebbero scoprire circostanze quali:

- L'opportunità di verificare preliminarmente in modo formale il dato di codice fiscale e partita iva da interrogare, onde evitare inutili interazioni con i sistemi informativi
- L'opportunità di restringere l'interrogazione anagrafica ai soli dati per la postalizzazione
- L'opportunità di vagliare la fattibilità, in alternativa, di una delle due seguenti soluzioni:
    - Inserimento dei dati postali acquisiti dall'anagrafe direttamente sulle lettere preconfezionate via RPA
    - Generazione delle lettere preconfezionate, comprensive dei dati di postalizzazione, con un minimo sforzo, quindi senza RPA

Quest'ultima soluzione, escluderebbe l'intervento della RPA e apparentemente sminuirebbe la portata della stessa. Invece, controintuitivamente, se questa soluzione andasse in porto, dopo opportune verifiche, sarebbe comunque merito degli analisti RPA che avrebbero giocato un ruolo strategico, facendo emergere contemporaneamente un problema sconosciuto ed una soluzione rapida, a portata di mano, della quale altrimenti nessun addetto ai lavori si sarebbe reso conto.

La **RPA-riduzione**

A prescindere dalla modalità con cui alla fine si sceglierà di automatizzare un'attività (non per forza quindi con RPA), la RPA può comunque tornare utile ad effettuare delle prove rapide per comprendere fino a che punto ci si possa spingere. Analogamente ad altri concetti matematici di riduzione, si può definire la **RPA-riduzione** come operazione di automatizzazione incrementale di un'attività. In altre parole, un'attività viene sempre più automatizzata fino al punto in cui si può dimostrare che è stata **RPA-ridotta** e non può pertanto esserlo ulteriormente. E' RPA-ridotta un'attività che non ha più nessuna componente manuale e non richiede quindi nessun intervento umano, nemmeno per la correzione di eventuali malfunzionamenti. "RPA-ridotta", però, non equivale ad "efficiente" ma ha comunque affinità con "efficace"; il tema dell'efficienza, o meglio, dell'ottimizzazione, come noto, è ben più complesso. Infine non è RPA-riducibile un'attività che ha limiti tecnologici estrinseci (ad esempio non avere le risorse umane o economiche necessarie a RPA-ridurre) o intrinseci (ad esempio quelle attività in cui ancora l'uomo è insostituibile).

Strumenti, framework e coordinamento

Classificazione e modalità d'uso

Andando alla sostanza della RPA, sono tanti gli strumenti e i framework che si possono considerare utili alla causa. Come nella maggior parte delle branche informatiche, anche nella RPA difficilmente può esistere uno strumento o un framework unico, onnicomprensivo, che copra tutte le casistiche (tante e tali sono infatti le modalità di interazione uomo-macchina, i programmi e gli ambienti con cui interagire). Inoltre, affidare tutto il funzionamento di un sistema ad un singolo framework, crea automaticamente un singolo punto di fallimento, ossia un piccolo anello della catena, il più debole, spezzato il quale non funziona più nulla. Un buon modello invece prevede che ci sia una eterogeneità di strumenti indipendenti (non necessariamente tra quelli a pagamento) in modo da assicurare minor impatto possibile a livello di sistema a seguito di un guasto e maggiore flessibilità possibile rispetto all'adeguatezza alle varie esigenze. Nella misura in cui ogni oggetto fa il proprio mestiere, senza travalicare i confini della precipua destinazione d'uso, si ottengono i risultati migliori. Viceversa, il pericolo che accomuna tutti i framework della galassia RPA è l'abuso degli stessi e la conseguente perdita di controllo.
Strumenti e framework per la RPA si possono classificare in vari modi. Tipicamente si hanno: Macro, EDM, framework visuali e non visuali. Discorso a sé andrà fatto per il sistema di coordinamento, controllo e monitoraggio di framework e strumenti, per il quale non esiste nulla di preconfezionato né tecniche consolidate.

Macro

Si tratta di una tipologia generica di strumento, di cui possono essere dotati alcuni software, che permette di registrare l'esecuzione di azioni ripetitive e/o richiamare tutta una serie di funzioni incluse in questi software attraverso dei comandi parametrici, anche in forma di script, quindi con possibilità di utilizzare tutti i costrutti dei linguaggi di scripting. Esempio tipico di Macro sono quelle incluse nel pacchetto MS Office (anche in forma di scripting con il VBA) ma ne esistono anche in altri contesti, quali ad esempio gli emulatori di terminali remoti (specie per i mainframe commerciali). In generale le Macro hanno diversi livelli di "potenza"; tramite il linguaggio VBA, ad esempio, è possibile andare ben oltre l'interazione con le funzionalità di tutta la suite MS Office.
I maggiori limiti delle Macro riguardano l'autoreferenzialità, la distribuzione e la sicurezza. L'autoreferenzialità è intesa come la limitazione di interazione delle Macro con il proprio ambiente specifico. Ad esempio VBA interagisce con tutte le funzioni di MS Office, con oggetti COM e librerie standard di MS Windows: è un perimetro notevole ma è anche un'eccezione nel mondo delle Macro. Tipicamente, infatti, le Macro interagiscono solo con il loro prodotto software di riferimento e spesso internamente a tale software. La distribuzione delle Macro presso gli utenti finali il più delle volte è limitata allo scambio di file aggiornati

con il contenuto (anche o solo) della Macro stessa, il tutto tramite email o copia/incolla o comunque tendenzialmente con modalità manuali. In alcuni casi è possibile costruire ad arte dei meccanismi di autoaggiornamento di tipo client/server, ad esempio con il VBA, ma si tratterebbe di un mascheramento del limite intrinseco delle Macro e richiederebbe lo stesso, in determinate situazioni, lo scambio manuale di file. Per quanto riguarda la sicurezza, ci sono vari aspetti limitanti, o ancor peggio, negativi quali ad esempio l'impossibilità di controllare accessi, permessi e codici sorgente di queste Macro ed inoltre la suscettibilità ad attacchi informatici o a controlli bloccanti da parte degli antivirus che possono interpretare alcuni tipi di file che includono Macro come potenzialmente pericolosi ed eliminarli dalla memoria.
In definitiva le Macro rappresentano, in ambito RPA, uno strumento ottimo per soluzioni il cui scopo sia accelerare alcune parti di lavori da effettuare all'interno dei prodotti specifici, per un singolo utente specifico e possibilmente invariati nel tempo. Le Macro sono un caso particolare di RPA ma hanno rappresentato e rappresentano tuttora una àncora di salvezza rapida e piuttosto efficace in contesti in cui non vi è la possibilità di investire su soluzioni informatiche gestionali onnicomprensive oppure in casi in cui si presentano ricorrenti manualità circoscritte ad un solo prodotto (come può essere MS Office) rimpiazzabili con questi semplici automatismi. L'azzardo e la tentazione di abusare delle Macro oltre gli opportuni confini (rapportati al contesto risorse/esigenze, rischi/benefici etc.), a lungo andare porta facilmente a problemi di manutenibilità, sicurezza, distribuibilità, correttezza e coerenza dei risultati finali.

EDM

Sono tutti quegli strumenti (Enterprise Data Manager) la cui principale funzione è quella di aggregare dati provenienti da varie fonti, soprattutto interne all'azienda, effettuando su di essi anche ulteriori elaborazioni. Possono prendere in input dati eterogenei e disgregati, purché intrinsecamente strutturati, provenienti da file (fogli di calcolo, CSV, TXT, PDF etc.), da database (MySQL, PostgreSQL, database commerciali, etc.), da web service e molto altro. Questi dati possono quindi essere combinati insieme e filtrati, rielaborati ed esportati in forme analoghe a quelle di input (su file, database, web service etc.). Questa caratteristica, ove possibile e praticabile, in alcuni casi riesce a coprire totalmente l'esigenza di una specifica automazione mentre in altri casi semplifica di molto le automazioni. La fase di acquisizione di dati in input, infatti, è un nodo cruciale che spesso in ambito RPA viene affrontato con metodologie di estrazione da maschere grafiche, con tutti i rischi e le difficoltà connesse. Quando invece si può utilizzare lo strumento di EDM questa fase viene rimpiazzata totalmente da una più stabile e sicura acquisizione che va direttamente alla fonte dei dati. Potendoli, addirittura, rielaborare è possibile iniziare e concludere un'intera automazione all'interno dell'EDM stesso (ad esempio quando non sono previste operazioni di tipo "dispositivo").
I pericoli principali degli EDM sono l'abuso e l'esplorazione in autonomia. A causa dell'abuso, quindi per un uso diverso dalle finalità per cui è sviluppato l'EDM, si perde il controllo di molti aspetti, specie all'inizio ed alla fine dell'esecuzione del compito. Ad esempio nella fase iniziale, se l'EDM è dotato di un proprio motore di schedulazione delle attività, usato correntemente per la RPA, ciò comporta l'onere di dover monitorare tutte le elaborazioni costantemente e con le modalità proprie di quel determinato prodotto. In

generale infatti è meglio rivolgersi al comparto IT ufficiale di schedulazione che lanci e monitori dall'esterno i processi dell'EDM, sia per questioni di presidio ufficiale, sia per questioni di omogeneità delle metodologie operative. Nella fase finale dell'elaborazione, ad esempio, per trattare i dati che sono stati estratti e combinati da varie fonti, spesso si ricorre all'utilizzo di linguaggi e procedimenti specifici dello stesso EDM. Anche in tal caso è opportuno, ove possibile, trattare i dati con strumenti più idonei, esterni all'EDM, al fine di separare i ruoli di ciascuno strumento ed avere uniformità operativa a livello di azienda, sia in fase di sviluppo che di manutenzione.
Il secondo pericolo riguarda l'esplorazione, che è da intendersi come la fase di analisi in cui si vanno a cercare le fonti più opportune di dati. Ad esempio, nel caso di sistemi informativi, si tratta di andare ad esplorare i db, gli schemi e le tabelle fino a trovare ciò che serve. Il problema è che in molti casi le fonti dei dati sono a supporto dei programmi e i programmi sono a supporto degli utenti. Quindi passare direttamente dalla fonte dei dati agli utenti è rischioso se non si conoscono le convenzioni e le logiche interne ai programmi, stabilite dai programmatori. Se l'esplorazione e la creazione di query ed estrattori simili viene fatta in autonomia, è alto il rischio che si estrapolino dati non corretti e/o non completi oppure che, anche se in quel momento i dati sono corretti, nel tempo la struttura dei dati venga modificata dagli sviluppatori dei programmi, senza preavviso per gli sviluppatori RPA, invalidando la stabilità dell'estrazione dei dati. Il rimedio è quello di coinvolgere attivamente i comparti di sviluppo chiedendo loro indicazioni specifiche su come estrapolare in tutta sicurezza i dati. Ad esempio, nel caso dei database, gli sviluppatori ufficiali potrebbero fornire l'accesso ad una "stored procedure" che rilascia i dati che servono alla RPA e della quale viene garantita nel tempo la manutenzione.

Framework visuali

I framework visuali sono dei prodotti, nati espressamente per la RPA, programmabili e gestibili quasi esclusivamente in modo grafico. La stragrande maggioranza di questi ha licenza commerciale e gira su sistema operativo Windows. Andare a descrivere in dettaglio cosa sono in grado di fare è abbastanza complicato ma in sostanza si prefiggono di automatizzare tutte le possibili interazioni di un utente con i programmi del computer, attraverso una serie di strumenti quasi sempre grafici e tendenzialmente semplici da usare per chi ha un minimo di dimestichezza con la programmazione; gli strumenti comuni più o meno a tutti i framework di questo tipo sono:

- l'ambiente di sviluppo - un ambiente predisposto per sviluppare l'automazione, attraverso la programmazione fatta in varie forme (come scripting proprietario o diagrammi a blocchi), anche con l'ausilio di oggetti preconfezionati, procedure guidate e utilità di vario genere, tra le quali quasi sempre si trovano:
    - il registratore, che permette di registrare le azioni compiute dall'utente sui programmi tramite i dispositivi di input (mouse, tastiera, etc.).
    - l'esploratore di interfaccia, che permette di analizzare e selezionare porzioni di interfaccia dei programmi e trasformarle in "oggetti" del formato del framework che si possono successivamente referenziare, manipolare, parametrare, riutilizzare etc.
- gli esecutori, che sono gli interpreti dell'automazione RPA che, installati su uno o più computer, eseguono l'automazione che è stata sviluppata

- il gestore centralizzato, che è un programma centrale, quindi lato server, che permette di coordinare gli esecutori (schedulando e monitorando le esecuzioni), memorizzare dati da elaborare, parametri e dati comuni a più automazioni (spesso chiamati “asset”), gestire i permessi, le versioni, la distribuzione, la sicurezza delle automazioni ed altro ancora

Il vantaggio di tali framework è che offrono questa gamma di strumenti e oggetti predisposti a gestire i casi d’uso più frequenti e importanti in ambito RPA, accelerando notevolmente lo sviluppo e la messa in produzione degli automatismi. Il potenziale svantaggio principale, come per tutti i prodotti commerciali, è il vincolo a schemi e convenzioni proprie di tali prodotti. La questione più spinosa e paradossale, però, è che questi framework stanno stretti un po’ a tutte le categorie di utilizzatori e nessuna di queste, singolarmente, riesce a beneficiarne al massimo della potenzialità, anche per i motivi che cerco di spiegare a seguire. I produttori di questi framework indirizzano le aziende nel farli adoperare ad analisti (o comunque non a sviluppatori IT), asserendo che siano alla loro portata. In molte aziende, però, questi prodotti prudenzialmente si danno in mano a sviluppatori IT, dato che c’è di mezzo anche della programmazione vera e propria. Di qui in avanti descriverò una serie di problemi relativi a questa diatriba sui framework visuali e, per semplicità, chiamerò “n.i.” (non informatici, quindi tipicamente analisti funzionali) e “i.” (informatici, quindi tipicamente sviluppatori IT) queste due categorie di addetti ai lavori della RPA.

Un i. si potrebbe imbattere nell’utilizzo di nuovi linguaggi di scripting oppure di schemi a blocchi di tipo “workflow“ che spesso limitano le capacità tecniche di chi vi deve operare. D’altro canto un n.i. potrebbe trarre vantaggio da alcune semplificazioni che in linea teorica permettono anche a chi non ha la dovute competenze di costruire automazioni. Ma il rovescio della medaglia è il seguente: tendenzialmente un n.i. affronterà lo sviluppo dell’automazione in modo semplicistico ed ottimistico, andando incontro a potenziali fallimenti in alcune circostanze, anche cruciali (per cui, ad esempio, non gestirebbe le eccezioni); un i., invece, avendo la professionalità di gestire le complessità e le eccezioni di un processo da costruire, potrebbe trovare difficoltà a barcamenarsi tra gli oggetti semplificati e schematici messi a disposizione dal framework. Altro svantaggio è che affidando ad un n.i. un framework visuale, proprio grazie alla dovizia di strumentazioni in dotazione, molte delle quali facili da usare, costui ne abusi quale surrogato di strumenti più idonei a determinati scopi. Altrettanto svantaggioso, infine, è considerare tali framework a prova di sostituibilità in corso d’opera della tipologia di sviluppatori. Nel caso in cui un’automazione sia stata sviluppata da n.i., infatti, un i. che subentri potrebbe dover rifare tutto se rileva gravi problemi di stabilità, correttezza e manutenibilità. Nel caso in cui un’automazione sia stata sviluppata da i., invece, questi potrebbero aver combinato in modo talmente articolato e complesso gli strumenti in dotazione che un n.i. subentrante non potrebbe mai gestirli ed anche un nuovo i. subentrante potrebbe avere difficoltà interpretative, proprio a causa dei predetti problemi di leggibilità e gestibilità.

Per arginare in parte le difficoltà fin qui elencate, si possono combinare vari approcci. L’approccio fondamentale è quello di stabilire regole e modelli. Come già detto, infatti, i framework visuali permettono con estrema semplicità di creare e combinare tra loro raffiche di oggetti preconfezionati e, di pari passo, generare molta confusione e discrepanze di struttura tra un’automazione e l’altra. Fissando invece dei modelli e degli schemi predefiniti, spesso proposti anche dagli stessi framework, e fissando anche regole sugli oggetti da prediligere, sulla modularità, su nomi e tipi delle variabili, sulla modalità di acquisizione e

scambio dati etc., si mette ordine in partenza al potenziale caos. Un altro approccio, specifico per gli i. ed applicabile solo nei framework espandibili, è quello di creare librerie ed estensioni ad hoc e riusabili, tramite linguaggi di programmazione compatibili, nel caso in cui il framework non disponga di un oggetto adeguato per una determinata operazione semplice o nel caso in cui si debbano combinare un notevole numero di oggetti semplici del framework per effettuare operazioni complesse (a scapito di linearità e leggibilità). Un ultimo approccio utile e sempre valido è quello di limitare l'uso del framework solo allo scopo per cui è nato. Ad esempio, nel caso in cui una automazione al proprio avvio richieda una scelta iniziale interattiva, in base alla quale si stabiliscono i parametri da usare nei passi successivi, non conviene abusare del framework per creare eventuali maschere o per far sì che tali scelte debbano effettuarsi tramite la compilazione di un foglio elettronico da parte dell'utente finale (con il rischio di errori od omissioni). E' più opportuno in tal caso creare un applicativo grafico che permetta di effettuare le scelte iniziali in modo guidato e controllato e dal quale scaturisca o l'avvio diretto dell'automazione (solitamente tramite un web service) o, tuttalpiù, l'esportazione di un file con i parametri da dare in input all'automazione.

Framework non visuali

I framework RPA non visuali sono quelli sui quali, di fatto, si basano i framework visuali. In altre parole, nella maggior parte dei casi, i framework visuali sono arricchimenti ed estensioni di Microsoft Ui Automation (nel caso dell'ambiente Windows). A loro volta, anche molti altri framework non visuali si basano su questo stesso framework non visuale. Si tratta in generale di librerie utilizzabili, spesso, tramite i più diffusi linguaggi di programmazione, quindi riservati solo a sviluppatori di professione. Vantaggi e svantaggi sono all'incirca speculari a quelli dei framework visuali. Lo svantaggio è di non avere oggetti riccamente predisposti che accelerino lo sviluppo ma solo oggetti di livello piuttosto basso che devono essere necessariamente combinati tra loro per poter ottenere risultati utili. Il vantaggio invece è quello che uno sviluppatore riesce ad avere pieno controllo e cognizione dei meccanismi che stanno dietro l'automazione dei quali invece, nei framework visuali, sarebbe all'oscuro e non potrebbe gestire appieno.
Questi framework non visuali sono nati a suo tempo per poter fare i test sulle interfacce e poi si sono diffusi anche per l'uso in ambito RPA ed in fin dei conti uno sviluppatore otterrebbe risultati più efficaci, efficienti ed affidabili con questi framework piuttosto che con quelli visuali. In ambiente Linux, peraltro, non esistono framework visuali e pertanto quelli non visuali risultano essere l'unica alternativa (https://ldtp.freedesktop.org/wiki/ e https://gitlab.com/dogtail/dogtail ne sono un esempio).
Probabilmente in grosse realtà aziendali tali framework non prendono molto piede, sebbene facciano risparmiare e, facendone buon uso, rendono di più di quelli visuali. Una possibile spiegazione è che, poiché tale categoria di framework è materia per professionisti, è più sensato - allargando la visione - che si impieghi un team di sviluppo professionale per l'informatica tradizionale piuttosto che per la RPA e di conseguenza si assegni la gestione di quest'ultima a n.i. oppure a sviluppatori meno d'esperienza. Ciò induce quindi ad optare per framework visuali che si pensa siano più alla loro portata ma la storia spiegata nel precedente paragrafo è un po' più lunga.

Problemi comuni a tutti i framework

Dalla panoramica finora fatta di questi framework, come già detto in premessa, si è sottolineato che la debolezza principale è l'uso improprio degli stessi. Talora si creano contesti in cui la RPA viene affidata in via esclusiva a dei non informatici oppure a degli informatici a cui viene fatta fretta oppure ancora viene adottato un solo framework con il quale si è costretti a fare tutto il possibile. Il risultato di questi ed altri scenari analoghi è che ci si arrangia come in una "cantina del fai da te" e non si ha il modo o il tempo o la capacità di elevare la visione e costruire una base solida di lavoro che tenda invece al modello "industria / catena di montaggio". Volendo sviscerare i problemi derivanti da uso improprio, o dall'anteporre pratica e rapidità ad analisi e metodo, si possono trattare vari aspetti:

- **Modelli, convenzioni e buone pratiche**. Proprio come è d'uopo nell'ambito dell'informatica tradizionale, buona norma vuole che ci sia metodicità e organizzazione nello sviluppo delle soluzioni RPA. Tale prassi passa innanzitutto dal creare a monte modelli e convenzioni, così da seguire una linea che rende più rapide tutte le fasi del ciclo produttivo: progetto, sviluppo e manutenzione. Va da sé che, a prescindere dal framework utilizzato, se ogni sviluppatore sceglie un proprio modus operandi di volta in volta, crea solo caos con il rischio di rendere ingestibili i prodotti. E' invece più corretto adottare innanzitutto dei modelli generali, che possono essere o forniti dal framework stesso, o suggeriti da fonti pubbliche nel web, o costruiti ex novo da utenti esperti. Tali modelli conterranno le definizioni dei parametri, delle fonti dei dati, dei metodi di gestione delle eccezioni, dell'interazione con l'utente, dei log, della sicurezza eccetera. Unitamente ai modelli (o all'interno degli stessi ove possibile) è opportuno stabilire anche convenzioni da seguire, ad esempio sui nomi delle variabili, sui tipi, sui nomi delle funzioni, su quali oggetti preconfezionati usare in base alle circostanze etc. Non bisogna trascurare infine le buone pratiche da seguire: anche se si tratta di RPA, considerata una branca dell'IT minore e marginale, strutturare le soluzioni in modo ottimale dà sempre vantaggi in termini di efficienza, leggibilità, manutenibilità, stabilità, correttezza etc. del prodotto.
- **Instabilità dell'interfaccia.** Come già detto in precedenza, spesso la RPA interagisce con interfacce che possono mutare nel tempo. Ogni framework reagisce ai cambiamenti a modo proprio. Se ci si aspetta di trovare un determinato elemento nello schermo che poi è sparito o mutato o addirittura è variato nella sostanza ma non nella forma (come quando un campo etichettato mantiene l'identificativo ma cambia il contenuto dell'etichetta associata), il programma di RPA si potrebbe arrestare del tutto oppure potrebbe fallire parzialmente oppure potrebbe andare avanti ignaro di tutto. Pur sapendo che questi ed altri casi analoghi non potranno mai essere gestiti correttamente al 100%, è buona norma, per tentare di mettere al riparo più possibile l'automazione, fare riferimento alla verifica degli elementi con cui si interagisce e delle loro etichette. Ad esempio, se c'è un campo etichettato da riempire, verificare sempre che testo e identificativo della rispettiva etichetta del campo, nonché l'eventuale testo e l'identificativo del campo stesso rimangano immutati nel tempo. Non guasta controllare anche che non mutino né qualitativamente né quantitativamente altri elementi chiave dell'interfaccia come titoli,

etichette di sezioni, numero di campi presenti etc. Altro problema di instabilità dell'interfaccia è il caso in cui questa sia stata sviluppata senza abbinare identificativi univoci o comunque senza criteri prestabiliti per gli elementi che la compongono. E' rischioso infatti sviluppare automazioni di interfacce che non contengono elementi identificabili, perché lo si farebbe attraverso metodi approssimativi come la posizione degli elementi sullo schermo (che dipende sempre da molteplici fattori che la rendono non assoluta) o l'ordine (dati tutti gli elementi dell'interfaccia, si seleziona l'ennesimo in ordine di apparizione ammesso che l'ordine non cambi mai). Con questi metodi, infatti, è più complicato verificare che gli elementi siano rimasti immutati nel tempo, per ovvi motivi, a meno di non provare a ricorrere all'intelligenza artificiale, ed in particolare alla computer vision, affinché identifichi le sezioni dello schermo con cui occorre interagire a partire dall'immagine della schermata stessa. Ma arrivati a questo punto non guasta ricordare che: tipicamente con la RPA si automatizzano altri software che, in quanto tali, avrebbero già dovuto automatizzare qualcosa ma non lo fanno, essendo stati evidentemente sviluppati in modo lacunoso, a cominciare dall'assenza di identificativi univoci degli elementi; pertanto si userebbe l'intelligenza artificiale, che ha comunque dei limiti e dei rischi intrinseci derivanti dal non determinismo, per interagire con un software lacunoso che porta quindi con sé ulteriori limiti e rischi. Posta così, generalmente la computer vision si auto-sconsiglia per questo caso d'uso.

- **Gestione, controllo e verifica.** In quest'ambito si trovano vari argomenti tra i quali, in particolare, il coordinamento tra più framework/ambienti, l'azionamento degli automatismi, la gestione della sicurezza e la registrazione dei log. Quando si mettono in piedi molteplici automazioni, la complessità di gestione delle stesse aumenta notevolmente, senza una misura precisa. Pensare di gestirle navigando a vista, oltre a rappresentare rischi concreti, può portare a vanificare ogni possibile beneficio della RPA. Ogni framework può avere delle modalità proprie con cui affronta le tre questioni in discussione, modalità che spesso possono aiutare mentre altre volte possono complicare ancora di più le cose. L'argomento è vasto ed elencare tutte le possibili combinazioni di problemi è arduo ma, giusto per rendere l'idea, segue una lista sommaria.
    - *Coordinamento tra framework*. Supponiamo di dover coordinare vari framework tra di loro, non solo di tipo RPA. Si può fare manualmente, ma prestando costante attenzione a non dimenticare uno o più passaggi nell'esatta sequenza. Si può fare automaticamente, ma ci dovrà essere un software che dovrà governare e orchestrare tutti gli altri, ammesso che questi siano controllabili in modo robusto, ad esempio tramite API o protocolli di scambio messaggi; se non sono così controllabili, invece, si dovrebbe ricorrere a metodi rudimentali come i file sentinella e gli inneschi continui delle automazioni, a vuoto fintanto che non si rispetti una condizione che faccia partire la vera e propria elaborazione etc. Questi metodi portano con sé ovvi problemi di efficienza, stabilità e sicurezza. Quindi, in definitiva, bisogna essere cauti nell'adottare framework che non favoriscano il coordinamento dall'esterno.
    - *Azionamento degli automatismi*. L'azionamento, innanzitutto, può essere di due tipi: frontend e backend. Il primo prevede l'intervento di un utente che

deliberatamente aziona l'automatismo e ne segue a vista l'esecuzione, occupandosi anche di gestire gli errori. L'altra modalità, backend, è l'azionamento automatico non assistito dall'utente ma, ciononostante, ci sarà sempre un utente che prima o poi dovrà controllarne l'esito. I problemi tipici della modalità frontend sono: la concorrenza tra utenti (chi deve fare partire l'automatismo, chi accede per primo ad un client per fare partire il proprio automatismo quando ci si contende un numero limitato di macchine RPA, etc.), la gestione degli errori (ognuno li "aggiusta" a modo suo), la regolarità (ci si può dimenticare di innescare l'automazione o la si può innescare più volte o in modo errato), il corretto inserimento dei parametri e dei dati in ingresso etc. I problemi tipici della modalità backend sono analoghi a questi ultimi ma da un punto di vista non più umano ma procedurale: come e quando fare partire un automatismo (schedularlo? farlo innescare dall'esterno?), stabilire un comportamento risolutivo di fronte ad errori o blocchi (senza banalmente avvisare l'utente e demandargli la sistemazione), riuscire a verificare in automatico che l'innesco avvenga regolarmente nei tempi e nei modi stabiliti, verificare che input e parametri dell'automatismo mantengano sempre il formato corretto etc. Su entrambe le categorie, front- e back-, i rischi di un cattivo innesco sono di spreco e/o saturazione delle risorse, mancata erogazione del servizio ed esecuzione errata o incompleta.

- *Gestione della sicurezza*. Una delle regole chiave della sicurezza informatica è quella di limitare i permessi di un utente ai soli ambiti di sua pertinenza. Questa stessa è bene che venga applicata agli utenti della RPA che peraltro possono essere di vario genere. O sono utenti reali, che usano le stesse credenziali con cui operano in altri ambiti, o sono utenze dedicate alla RPA ma sempre associate ad utenti reali (finalizzate a distinguere l'operatività normale che compie un utente da quella gestita tramite una procedura RPA), o infine sono utenze robotiche, quindi di tipo applicativo e non personale. Queste ultime, c'è da chiedersi anche, a nome di chi operano? In certi ambiti aziendali, specie ove c'è responsabilità diretta dettata dalla normativa, è una tematica cruciale. Tutte queste utenze, poi, devono essere impostate in modo da evitare il predetto rischio di dare troppi permessi e, nel caso di utenze robotiche innescate direttamente o indirettamente da utenti reali, deve esserci comunque una traccia inequivocabile di chi ha fatto cosa. Districarsi tra queste molteplici attenzioni è tutt'altro che semplice e la situazione si complica ulteriormente se, oltre ai permessi applicativi, si mettono in ballo i permessi di accesso a file system per via di utilizzi di file di parametri, log, input/output di dati dell'elaborazione. Inoltre c'è anche la questione dell'accesso alle stesse procedure RPA che, se non si presta la dovuta cautela, specie nei framework visuali con innesco frontend, è facile che siano visibili ed avviabili da tutti gli utenti, compresi quelli non interessati. Altro problema, subdolo, di alcuni inneschi frontend è che gli utenti più esperti potrebbero alterare il codice sorgente della procedura RPA (spesso sono linguaggi proprietari interpretati e quindi i file sono accessibili) ed eseguire azioni malevole. Per via di queste ultime considerazioni, in generale è meglio usare solo gli inneschi backend.

    - *I log*. È importante che si sappia chi ha fatto cosa e, innanzitutto, quale utente abbia eseguito quale procedura RPA. Molti framework registrano in automatico questa informazione ma, come già accennato, nel caso di un innesco indiretto (utente reale che avvia automazione con utente robotico) l'informazione potrebbe andare persa. Inoltre è opportuno memorizzare costantemente il punto dell'elaborazione in cui si è arrivati al fine di poter riprendere da lì in caso di arresto improvviso (sebbene la gestione degli errori, fatta in modo corretto, prevenga la maggior parte di queste evenienze); è anche opportuno memorizzare record per record l'esito dell'elaborazione ed infine, in caso di cattura di errore (catch), stampare anche un print screen a corredo di tutte le altre informazioni tecniche che si memorizzano tipicamente. Questo può aiutare il programmatore a capire ancora meglio la causa dell'errore. Attenzione infine a prevedere i log su semplici file: potrebbe voler dire che gli utenti esecutori saranno in grado di manipolarli ed anche far sparire tracce importanti.
- **Modularità e riuso**. Una casa produttrice di software può avere tipicamente un numero esiguo di soluzioni in vendita e, per ciascuna di queste, organizza il codice sorgente per garantire modularità e riutilizzo. In ambito RPA, invece, è facile ritrovarsi nella situazione opposta: non essere una casa di produzione software ed avere un numero enorme di soluzioni prodotte (ciascuna relativamente piccola). In questo scenario la modularità potrebbe complicarsi sotto vari aspetti: le automazioni il più delle volte sono programmi di piccola grandezza, da sviluppare in fretta e in grande quantità, quindi è facile trascurare l'aspetto della modularità; molti framework, specie quelli visuali, hanno una sorta di linguaggio di programmazione che può essere modularizzato in modo molto limitato per svariate ragioni (assenza di programmazione ad oggetti o assenza di costrutti analoghi alle funzioni o altro); se un modulo è collegato ad un'automazione tramite il riferimento assoluto ad un file sorgente (ad esempio con la funzionalità "asset" tipica della RPA), la modifica di questo file potrebbe aggiustare un malfunzionamento su un processo e contemporaneamente crearne di nuovi su altri processi che usano lo stesso modulo. Di frequente si usa modularizzare parti comuni non solo ad una stessa automazione ma soprattutto a più automazioni (basta immaginare ad esempio che molti software con i quali interagire, pur essendo diversi tra loro, abbiano in comune la parte di autenticazione e selezione di un ambiente di lavoro). Purtroppo però è altrettanto frequente che avvengano le già citate modifiche ai software automatizzati, all'insaputa del programmatore RPA, il quale di punto in bianco potrebbe trovarsi a dovere modificare moltissime automazioni o un modulo in comune a più automazioni. Nel caso di questo modulo comune, per i problemi già detti, è sempre bene rifare l'intero test di ciascuna singola automazione coinvolta, onde evitare danni collaterali inaspettati.
- **Backup e diversificazione**. Le infrastrutture relative ai framework devono essere sottoposte a backup. Quindi tipicamente, essendo client e server, ciascuno dei due componenti deve essere duplicato e deve anche avere delle modalità di ripristino a versioni precedenti. Spesso questo aspetto viene trascurato, data la natura temporanea della RPA ma, considerato che la temporaneità spesso si tramuta in perennità, il tassello "backup" diventa fondamentale. Ciò anche per un altro motivo:

di frequente si adotta uno e un solo framework ed una e una sola infrastruttura IT relativa ad esso. Proprio per questa ragione, ove possibile, oltre al backup è importante diversificare i framework adottati. Se si blocca un framework non è detto che ripristinando il backup questo possa ripartire facilmente. Pertanto quella parte di automazioni che sono state sviluppate con altri framework, quantomeno continueranno a funzionare indipendentemente. I costi delle licenze dei framework commerciali di solito sono impegnativi, specie per quelli visuali, ma è anche vero che ci sono strumenti, non per forza visuali, che sono anche gratuiti o molto meno costosi e vale la pena adottarli per diversificare il comparto.

- **Flessibilità ed espandibilità**. Questi temi riguardano i limiti dei framework, in particolare limiti operativi e limiti di funzionalità. I limiti operativi il più delle volte si trovano nei framework visuali, i quali impongono modalità rigide con le quali programmare le automazioni. Ad esempio alcuni prevedono la programmazione con linguaggi di scripting proprietari misti a procedure guidate di generazione del codice. Altri propongono parametrizzazioni di avvio e gestione dei processi centralizzati solamente tramite apposite interfacce grafiche. Altri impongono una sorta di programmazione semplificata dei processi tramite “workflow” grafici ossia in cui i costrutti tipici della programmazione (“assegnamento”, “iterazione”, “condizione” etc.) e le funzionalità predefinite si disegnano sullo schermo con forme geometriche (quasi sempre rettangoli) interconnesse tra loro. Il più delle volte, nelle intenzioni delle case software, c’è l’obiettivo di semplificare la programmazione delle procedure tanto da renderla alla portata anche di chi non sa programmare, con le conseguenze però di cui si è già parlato. Tutto ciò considerato, sarebbe bene scegliere, ove esistenti, framework che offrano la maggiore flessibilità possibile in termini di utilizzo e gestione, attraverso modalità ibride. Ad esempio che diano sia strumenti visuali (per la rapidità) ma anche il corrispettivo in API, con la possibilità di esportare la parte visuale in codice puro e/o di scegliere il linguaggio con cui utilizzare le API attraverso apposite librerie. Allo stesso modo che fornisca API (ad esempio di tipo *opendata*) pure per le funzionalità a latere dell’automazione, come la sicurezza, la schedulazione, la parametrizzazione etc. Parlando invece del secondo aspetto in questione, lo si può riassumere nel fatto che i framework, come tutti i prodotti software, vengono implementati ed aggiornati periodicamente con nuove funzionalità, anche per andare incontro alle richieste della clientela, oltre che alle innovazioni tecnologiche. Tuttavia non è detto che le funzionalità, vecchie o nuove che siano, rispondano perfettamente alle esigenze degli utenti. Pertanto un framework RPA che sia espandibile liberamente offre notevoli vantaggi. Il più importante di questi è che, creando una libreria di funzionalità indipendenti dal framework, queste si possono poi collegare liberamente a uno o più framework RPA, o ad un nuovo framework da adottare in futuro, nel caso di rimpiazzo per qualsiasi ragione, anche economica.

Ottenere e gestire la sostenibilità

Per far sì che la RPA funzioni al meglio e dia frutti concreti e stabili, esistono tanti validi approcci. Occorre fare innanzitutto distinzione a proposito del contesto di applicazione e, in particolar modo, a due fattori: la RPA-favorevolezza e la quantità di risorse umane e materiali a disposizione.
Se ad esempio il comparto RPA dispone di un unico addetto a cui viene dato in dotazione un unico framework e a cui viene affidato l'incarico di automatizzare procedure RPA-non favorevoli, verosimilmente si può solo suggerire di "evitare errori grossolani" e non si può pretendere di applicare tutte le possibili accortezze per lavorare bene. Condizione necessaria per ottenere la sostenibilità è sicuramente avere abbastanza risorse umane e materiali. Ciò premesso, i paragrafi seguenti riportano una serie di accorgimenti, legati ciascuno ad un aspetto diverso dell'applicazione della RPA, la cui attuabilità è commisurata alla quantità di risorse a disposizione.

Accorgimenti suggeriti per la RPA

La squadra di lavoro

Pur non esistendo una squadra di lavoro ideale, ci sono sicuramente degli aspetti da valutare con molta attenzione per costruire un gruppo che porti a buoni risultati.

1) *La specializzazione*. Il campo della RPA è molto particolare e ostico dal punto di vista dell'applicabilità. Per evitare delle soluzioni frettolose o fallimentari, occorre che gli addetti siano molto prudenti ed accorti nell'approcciarsi all'analisi ed alla risoluzione dei problemi. Per far questo occorre che, se non sono già "esperti", si riservino di effettuare lunghi e articolati approfondimenti tecnici in fase di analisi. Quando si è già esperti ci si sa muovere molto meglio e si sa scegliere a quali realizzazioni rinunciare per dedicarsi ad altre che avranno più possibilità di efficacia complessiva. Poiché la stragrande maggioranza delle "variabili" di un progetto RPA non sono sotto il diretto controllo di analisti e programmatori, capitano spesso casi come questi:
    a) si porta in avanzato stato realizzativo artefatti mal concepiti in fase di analisi; un tassello mancante (come ad esempio un'informazione mai verificabile o mai reperibile), del quale non si era fatto caso inizialmente ma ci si accorge solo tardi, può invalidare e far fallire l'intero progetto (sia da intendere: a quel punto non sarà da rifare ma proprio irrealizzabile)
    b) si realizza un artefatto che non dà risultati stabili nell'interazione con le interfacce grafiche, perché non ci sono casistiche fisse o per mancanza di etichettature o altro
    c) si porta in avanzato stato realizzativo un artefatto che non tiene conto di una determinata casistica non prevista in fase di analisi; se la si implementa successivamente, si corre il rischio di stravolgere molto del lavoro già effettuato; se viene fuori anche una seconda casistica non prevista all'inizio,

esperienza insegna che di lì in avanti se ne possano scoprire di ulteriori e valga la pena rivalutare complessivamente costi e benefici dell'intervento

Pertanto, nel gruppo di lavoro occorre che ci sia almeno uno specialista RPA da coinvolgere in tutti i progetti nelle fasi cruciali, specie quelle iniziali. Il più delle volte, infatti, è in grado di intuire i punti deboli degli interventi richiesti durante l'analisi e prevenire inutili sprechi di tempo e risorse su interventi che non andranno in porto.

2) *La qualifica*. In alcuni dei precedenti paragrafi dell'articolo ho citato la distinzione informatici / non informatici (i. / n.i.). Se ci si sofferma sugli aspetti umani di tale distinzione, in alcuni contesti lavorativi di RPA potrebbe sembrare che si possa urtare la suscettibilità dei n.i. in quanto "non qualcosa". In realtà anche gli i. potenzialmente si potrebbero sentire dei n.i. se chiamati a svolgere compiti nei quali il valore aggiunto della loro qualifica conta molto poco e, purtroppo, in ambito RPA questa condizione non è inverosimile.

   In realtà, se ben organizzato, il gruppo di lavoro deve avere un buon bilanciamento di entrambi i ruoli ed inoltre i n.i. devono comunque avere le competenze informatiche minime, tali per cui al cospetto del mondo esterno al gruppo RPA, appariranno abbastanza "informatici" anche loro che tecnicamente non lo sono. Per di più il bilanciamento ottimale richiede una netta prevalenza numerica dei n.i.. Se il gruppo di lavoro è molto piccolo, ad esempio di due unità, se entrambi sono dei n.i. in gamba possono ottenere dei risultati accettabili. L'aggiunta di un i. ai n.i. deve apportare dei benefici il cui risultato finale è quello di moltiplicare la resa del lavoro del gruppo; in altre parole un gruppo di 2 n.i., aggiungendo un i., deve produrre gli stessi risultati che avrebbero prodotto 6 n.i., giocando opportunamente sulla suddivisione dei compiti.

3) *I compiti*. Si considerino due scenari, uno privo di i. e un altro che li include. Nello scenario senza i., i n.i. dovranno operare normalmente ma con l'unica accortezza di non assumere impegni né effettuare valutazioni tecnicamente insostenibili per un n.i.. Nel caso in cui nel gruppo RPA vi sia almeno un i., i compiti si dovranno suddividere opportunamente. Segue un possibile schema.

   I n.i. più esperti in RPA potranno:

   a) supervisionare analisi e sviluppo
   b) presidiare l'evoluzione degli schemi operativi adottati
   c) studiare gli strumenti in dotazione e aggiornarsi
   d) interfacciarsi con gli i.

   I n.i. meno esperti (ma anche i più esperti) potranno:

   a) effettuare il lavoro ordinario (analisi, sviluppo, test etc.)
   b) curare la manutenzione e la documentazione

   Oltre a poter svolgere, ove si renda necessario, tutti i compiti previsti per i n.i. (anche talvolta con meno bravura), gli i. hanno dei compiti peculiari e strategici grazie ai quali si può incrementare la produttività del gruppo:

   a) curare installazione, aggiornamento, modalità operative tecniche legate ai vari software adottati (prevenendo futuri problemi che assorbirebbero energie di altri comparti IT o del gruppo RPA stesso)
   b) sviluppare moduli, modelli preconfezionati e funzionalità aggiuntive ai software adottati (ciò, se ben fatto, velocizza tutte le fasi del lavoro)

c) sviluppare software complementare a quello RPA ufficiale che semplifichi alcune fasi cruciali come acquisizione ed esportazione dei dati
d) guidare la strategia di sviluppo impartendo ai n.i. importanti nozioni tecniche
e) gestire aspetti complessi come la sicurezza, l'integrità e l'efficienza di software e dati, che diversamente potrebbero essere trascurati
f) coadiuvare i n.i. nell'affrontare la creazione di procedure RPA di particolare complessità e difficoltà tecnica

L'analisi

Nell'articolo vi è già un paragrafo interamente dedicato all'analisi e si può fare riferimento a quello. L'unica precisazione in più da effettuare riguarda il contesto della squadra di lavoro per come la si è definita poc'anzi. Se nella squadra sono presenti gli i., occorre coinvolgerli attivamente (con una qualsiasi modalità) anche durante la fase di analisi proprio perché sono quei soggetti tipicamente in grado di dare quei vantaggi di cui si è già discusso, tra i quali la prevenzione di errori tecnici e di usi impropri della RPA. Sarebbe forte la tentazione di lasciare agli i. solo i compiti più complessi e tecnici ma la fase di analisi è il cuore stesso della RPA e pertanto deve vederli direttamente coinvolti.

La documentazione

Dato che la RPA è provvisoria è bene che abbia una documentazione snella ed essenziale, che riporti solo quegli elementi utili a riprendere dopo tempo e rammentare i punti cardine di un'automazione. Pertanto immagino la documentazione alla stregua di una scheda sintetica, come un modulo da riempire, che riporti quantomeno:

- un nome breve e romanzato per l'automazione (ad esempio, nel caso dell'automazione esemplificativa descritta nel paragrafo dedicato all'analisi, si potrebbe chiamare "La Spedizione"). Per esperienza dico che i nomi dall'aria letteraria hanno in sé la capacità di stimolare i ricordi, di dire tanto con poco
- una descrizione sommaria di ciò che fa l'automazione con annesso possibilmente uno schema essenziale e rappresentativo delle fasi salienti del processo
- l'elenco dei framework coinvolti, sia quelli sui quali è sviluppata l'automazione sia quelli coi quali l'automazione interagisce, abbinati a ogni fase del processo
- la modalità di innesco (assistita / non assistita e, nel caso sia assistita, specificare anche come effettuare l'avvio mentre, nel caso non sia assistita, specificare anche il criterio di innesco come ad esempio polling, pianificazione, trigger etc.)
- utenze coinvolte, tipologie (robotiche o no) e a quale titolo il loro coinvolgimento
- elenco e posizione dei file sorgente, sia di sviluppo/test che di produzione
- elenco e posizione delle risorse utilizzate (uri di file di interscambio, nomi di database e tabelle, indirizzi web di applicativi esterni)
- nomi e recapiti dei referenti coinvolti, sia interni al gruppo RPA che esterni

I framework

Per ciò che riguarda i framework bisogna fare i conti innanzitutto con le singole realtà aziendali e poi rapportare tutto all'ampia dissertazione presente nella precedente sezione ad essi dedicata. Se un'azienda ha pochi fondi, potrebbe avere un solo addetto alla RPA e

possibilmente non poter acquistare nessun framework. Progressivamente, se si hanno a disposizione maggiori fondi si potrebbe acquistare un framework e/o ampliare l'organico con un ulteriore addetto, e così via crescendo. Tipicamente le aziende che hanno pochi fondi tendono a razionalizzare le scelte e possono anche trovare in un addetto i. e nell'assenza di specifici framework commerciali comunque un valido aiuto, dato che si può sempre ricorrere a framework gratuiti.
Il vero problema, a mio avviso, è quando si hanno a disposizione molti o moltissimi fondi e si corre il rischio di conseguenza di effettuare acquisti poco oculati (sia in termini di risorse umane che in termini di framework commerciali). Ad esempio con molti fondi si può decidere di acquistare un framework che sulla carta viene pubblicizzato come completo e potente ma che poi, come d'altronde tutti i prodotti del mondo, si scopre avere determinati limiti, anche nei punti di forza. Se poi è costato tanto, spesso gli addetti RPA sono costretti a giustificarne la spesa facendone un uso diffuso, al limite con l'abuso. Se invece si hanno a disposizione ancor più che molti fondi, si potrebbe iniziare a spendere e spandere qua e là in framework commerciali che diventa complicato gestire e far interagire tra loro o dei quali non è ben chiaro che uso fare, generando di conseguenza molta confusione.
Allora, tirando le somme, se si portano in squadra degli addetti RPA di esperienza, questi potranno essere consultati per dare i consigli giusti, in rapporto agli obiettivi che si pone l'azienda, rispetto agli eventuali acquisti da fare. Altrimenti è sempre bene fare delle prove assolutamente compiute ed in buon numero prima di effettuare acquisti ed è fondamentale che le prove siano condotte dal team RPA e non da altre strutture aziendali che hanno diverse mansioni. Per prove compiute e in buon numero, intendo tre o più automazioni concrete da realizzare, basate su reali esigenze, che differiscano più possibile tra loro su tutti i fronti. Nel dettaglio si potranno confrontare tra loro le varie caratteristiche dei framework testati (rapidità di sviluppo, modalità di pubblicazione e di esecuzione, stabilità, sicurezza, espandibilità, controllabilità tramite API, etc.) e tirare le somme, anche rispetto al rapporto qualità/prezzo e alle esigenze aziendali (ad esempio se l'azienda predilige la RPA come soluzione temporanea va bene anche risparmiare). La scelta dei framework deve essere sempre approfondita con i giusti modi e tempi affinché diventino un valido supporto al lavoro della squadra e non un ostacolo e un ulteriore problema costante da risolvere.

Sviluppo, test e produzione

Il ciclo di produzione di un applicativo RPA sembrerebbe sposarsi perfettamente con l'ormai famigerata modalità "agile". Ma paradossalmente anche un tale sistema snello potrebbe complicare inutilmente le cose. Partendo dalla tesi cardine di tutto questo articolo, ossia che la RPA è provvisoria, per fruire al massimo dei suoi vantaggi occorre che si faccia uno sviluppo con un'altra modalità, che possiamo definire **fragile**. Il ciclo di produzione deve essere ridotto all'osso perché l'applicazione della RPA deve essere rapidissima e circoscritta alla risoluzione di problemi quanto più atomici possibili, evitando di prendere l'incarico di applicarla a macro-processi complessi e articolati (ad alto rischio di fallimento, per automatizzare i quali potrebbero non esistere linee guida e buone pratiche che mitighino questo rischio).
Quindi, ricapitolando, il presupposto è che la RPA sia sempre provvisoria e risolva problemi di piccola entità o al più, ove RPA-favorevoli, media entità. A queste condizioni, fatta l'analisi e stabilito un ristretto gruppo di lavoro (in media 1 o 2 addetti), un buon approccio allo

sviluppo comprende le seguenti fasi: riepilogare i punti da sviluppare (anche su un pezzo di carta), svilupparli rapidamente, testarli alla svelta con l'aiuto dell'utente, fare le dovute correzioni di corsa, provare al volo con gli utenti la versione definitiva e mettere tutto in produzione. Non occorre scendere ulteriormente in dettaglio sui singoli punti di questo schema. In pochi giorni lo sviluppo "fragile" può essere concluso, portandosi addosso tutte le sue peculiari fragilità. Ciò detto, come si può essere così tanto rapidi? E come si può non temere le fragilità?

Si può essere rapidi perché la RPA è provvisoria, perché il processo da automatizzare deve essere relativamente semplice e soprattutto perché **la fase di analisi deve durare molto più di quella di sviluppo**, permettendo di delineare in anticipo e con precisione tutto il da farsi. E' un po' come quando nei film i ladri pianificano la rapina perfetta o i truffatori la truffa perfetta, studiando a menadito tecniche e tecnologie da usare, movimenti, insidie, punti deboli, percorsi migliori etc., e a quel punto truffare e rapinare diventa una passeggiata. Lo stesso vale in ambito RPA allorquando si agisce preliminarmente in modo opportuno sulla procedura da realizzare.

Si può non temere le fragilità se si segue un buon **modello di sviluppo** e un buon **modello di controllo** dei processi in produzione. Per quest'ultimo si può adottare un qualsiasi sistema, anche "fatto in casa", che accentri la gestione di tutti i processi RPA messi in piedi e faccia buon uso di metriche e log (quindi si rimanda sia ai precedenti argomenti discussi che, ad esempio, alla proposta di gestionale RPA che viene descritta più in avanti). Per il modello di sviluppo invece occorre che la squadra RPA ai suoi albori, quindi subito dopo essere stata messa in piedi, faccia un lavoro preliminare svolto in parte prima di accettare qualsiasi incarico e in parte, per successivo affinamento, durante lo sviluppo dei primi processi di automatizzazione. Ciò che occorre fare è definire vari punti, come ad esempio:

- stili di sviluppo da adottare (modelli preconfezionati, modelli creati ad hoc, moduli, librerie etc.), anche in riferimento a ciascun framework a disposizione
- convenzioni sulla nomenclatura degli oggetti utilizzati (file sorgenti, directory, risorse, variabili, funzioni, moduli etc.)
- stati basilari dei processi e delle singole lavorazioni dei processi
- ubicazione e modalità di immissione degli input, modalità di conservazione e consultazione di dati di input e output (db, filesystem etc.)
- modalità di utilizzo dei commenti, al fine di ridurli al minimo (ad esempio farlo solo ove si sia dovuto deviare dalle convenzioni stabilite o ove il processo sia più complesso e meno leggibile)
- repository da usare e modalità di backup (git, nas con backup, etc.)
- modalità di deploy dei processi (con script, con copia-incolla, etc.)
- modalità di generazione e conservazione dei log
- modalità di gestione di permessi e sicurezza

Una volta fatto lo sforzo iniziale, necessariamente rallentante, di stabilire questa traccia e poi collaudarla e affinarla durante lo sviluppo dei primi automatismi, ecco che la velocità di sviluppo, test e produzione viene assicurata. Le fragilità sono pienamente sotto controllo perché si sa già, qualora ci fosse un problema, dove e come andare a indagare e risolverlo grazie all'applicazione dei modelli di sviluppo e di controllo.

La manutenzione

Forse, in generale, parlare di manutenzione in ambito software è un argomento improprio. Un'apparecchiatura meccanica o elettronica, che quindi segue le leggi della fisica e interagisce con l'ambiente, è normale che si usuri o si guasti e con la manutenzione poi venga ripristinata. Un software, in quanto bene immateriale, non si dovrebbe mai né usurare né guastare. Quindi quella che viene chiamata "manutenzione", per il software tipicamente non è altro che la riparazione di problemi conseguenti ad un'errata progettazione o realizzazione. Ad esempio se si esaurisce lo spazio di memoria evidentemente non si è calcolato in modo adeguato il consumo di risorse. In ambito RPA tutti possibili errori di progettazione che richiedono la manutenzione possono essere quasi del tutto evitati se si effettuano per bene le fasi di analisi e sviluppo. Inoltre, però, la manutenzione subentra se viene meno la provvisorietà dell'automatismo e quindi, nel tempo, qualora dovesse essere modificato l'ambiente con cui l'applicativo interagisce, si rischia che lo stesso non funzioni più. Per cui si tratta di scelte delicate. Se l'azienda è consapevole che il processo "interagito" è rimasto immutato da decenni, si può fare la scommessa che il processo RPA "interagente" possa vivere altrettanto a lungo senza incidenti di percorso. Altrimenti la manutenzione sarà sempre dietro l'angolo. Inoltre, poiché come già detto il processo RPA per essere vantaggioso economicamente e operativamente viene sviluppato in modalità "fragile", è sempre l'ennesimo di una lunga serie che vengono sviluppati con grande frequenza e con documentazione sommaria. Per cui a distanza di tempo può diventare ostico e oneroso andare a ricordare l'analisi e comprendere come e dove effettuare le opportune modifiche. Soprattutto perché, a differenza delle procedure "interagenti", all'interno delle quali è facile orientarsi grazie ai modelli di sviluppo e di controllo, le procedure "interagite" sono praticamente tutte diverse tra loro e, in caso di manutenzione, devono essere ri-analizzate per filo e per segno.
Con questo argomento, si ritiene di aver elencato motivi a sufficienza affinché la RPA rimanga uno strumento prezioso solo nel breve periodo, cioè rimanga, appunto, sempre e solo provvisoria e venga rimpiazzata prima possibile da un altro genere di software.

Un gestionale RPA

Una grande azienda che voglia beneficiare dei vantaggi della RPA, deve adottare oltre al modello di sviluppo, anche quello di controllo, atto a prevenire più possibile i problemi fin qui discussi. In particolare, **limitatamente ai problemi elencati nel paragrafo "gestione, controllo e verifica"** della sezione "problemi comuni a tutti i framework", potrebbe essere d'aiuto costruire uno strumento modulare e flessibile, indipendente dai framework e contemporaneamente compatibile con la maggior parte di essi. Si descrive di seguito per sommi capi, giusto per dare un'idea, un possibile artefatto del genere, frutto di prove e applicazioni sul campo che hanno dato buoni risultati.

Architettura

L'idea si basa sulla creazione di un applicativo di gestione della RPA client-server, con un web-server e un database, che offra funzionalità in modo passivo a mezzo API ed in modo attivo tramite interfaccia web. Il principio di passività serve a preservare la peculiare natura RPA dell'applicativo, di modo che non si prenda la briga di sostituirsi ad altre funzioni aziendali come la schedulazione e la sicurezza.

Il database

Il database, di tipo relazionale, viene strutturato avendo come tabelle di base le risorse da gestire, tutte contraddistinte da una chiave univoca per tabella, ed in particolar modo:

- I framework. L'elenco dei framework utilizzati in azienda, con una categoria che li contraddistingua (visuale, non visuale, EDM, Macro etc.).
- I processi. L'elenco e la descrizione delle macro-procedure create, da associare ad uno o più automatismi. Questa tabella contiene anche campi speciali per configurare in modo programmatico alcuni meccanismi a livello di singolo processo.
- Gli automatismi. L'elenco dei singoli automatismi creati, ognuno dei quali associato ad un solo framework, che compongono un processo. Per meglio chiarire, ad esempio, per un ipotetico processo "paghe" ci saranno tre automatismi: "estrai lista" sul framework EDM, "inserisci lista" sul framework visuale e "stampa lista" su una MACRO.
- Le macchine. L'elenco e la descrizione degli host coinvolti, associati al/ai rispettivi framework.
- Gli utenti. Gli username degli utenti coinvolti nei processi.
- Le pratiche. Sono dei macro insiemi di record di dati da far lavorare ai processi (i raggruppamenti possono seguire logiche arbitrarie, come ad esempio: per giornata lavorativa, per nominativo di un cliente, per richiesta di un utente etc.). Questi macro insiemi hanno associato un unico processo.
- I dettagli. Sono i singoli record che compongono le pratiche, quindi ciascuno associato ad una e una sola pratica e ognuno con due diversi stati. Il primo stato è quello più esterno, utile all'utente finale per capire l'avanzamento, e rappresenta il punto del processo in cui un dettaglio è arrivato (ad esempio "fase 2 - inserimento elenco"). Il secondo stato invece è interno, ad uso e consumo degli automatismi al fine di tracciare a che punto è arrivata la singola fase di elaborazione. Sia alle pratiche che ai dettagli possono essere associati un'utenza corrente ed un timestamp dell'ultima azione compiuta, di modo da poter gestire la concorrenza e il carico di lavoro tra gli utenti.
- I permessi. La tabella che associa gli utenti ai processi e/o agli automatismi, di modo da garantire massima granularità sulla divisione dei compiti. Serve anche a censire gli utenti amministratori del gestionale.
- I log. L'elenco degli eventi registrati sia dal gestionale che dai vari automatismi, tramite le API del gestionale. In primis quale utente reale ha fatto cosa e quando.
- API esterne. Elenco di API e relativi parametri, associati ai processi, richiamabili direttamente dal gestionale ed associabili anche allo stato di una pratica.

L’interfaccia web

Serve agli amministratori per configurare il gestionale in modo visuale e agli utenti finali per interagire con i processi, orientandosi a colpo d’occhio sullo stato di avanzamento delle pratiche. Le caratteristiche più importanti sono due:

- Il diagramma degli stati. Dall’interfaccia gli amministratori possono configurare i processi di modo che al raggiungimento di un determinato stato interno, al singolo dettaglio venga attribuito in automatico un determinato stato esterno. Questo fa sì che sia gli utenti che i framework coinvolti in un processo, possano elaborare i record in modo corretto e controllato centralmente, senza sbagliare la sequenza degli automatismi da eseguire.
- La gestione delle pratiche. Gli utenti finali hanno un unico strumento per gestire l’avanzamento delle pratiche, che gli permette non solo di sapere se qualcosa è andata a buon fine oppure no, il perché e a che punto si trova, ma anche di effettuare verifiche tra una fase e l’altra di un processo. Ove, ad esempio, sia richiesta una modifica manuale ai dati dei dettagli o una verifica umana (non automatizzabile) prima di passare alla fase successiva, l’utente potrà effettuarla e poi decidere con un apposito bottone di far andare avanti o meno l’elaborazione, anche previo aggiustamento manuale del contenuto dei record.

API e funzionalità

Ogni funzionalità del gestionale deve avere una corrispettiva API. Questo perché si possa integrare con le varie funzioni aziendali specializzate (sicurezza, schedulazione etc.). Principalmente le API mascherano l’interazione con il database e quindi si occupano in sostanza di inserire e modificare i dati. Le uniche particolarità sono il controllo dei dati e l’avanzamento del diagramma degli stati. Per controllo dei dati si intende che, da interfaccia web, si può configurare il tipo di dato dei record da lavorare di modo che, qualora sia prevista l’interazione con gli utenti che possono modificare lungo il percorso il contenuto dei dettagli delle lavorazioni, avvenga un controllo sulla correttezza del dato inserito o modificato. Per avanzamento del diagramma degli stati, si intende la modifica automatica dello stato esterno di un dettaglio e dello stato di una pratica, in base all’esito. Ad esempio l’amministratore può aver configurato un processo affinché se si è nello stato esterno “fase 1” e si passa allo stato interno “ok”, lo stato esterno diventi “fase 2”. Il tutto sta dietro all’API “modifica stato dettaglio”. Più in generale, segue un elenco di API/funzionalità del gestionale a titolo esemplificativo:

- Inserisci/modifica/interroga processi/pratiche/dettagli/utenti/permessi/log
- Cambia stato dettaglio/pratica (che scatena eventi automatici come la modifica di stati esterni e/o l’innesco di API esterne)
- Richiama API esterna al gestionale (come l’innesco di una elaborazione)
- Avvisa utenti (si può configurare un sistema di avvisi - tramite chat/email/altro - per varie casistiche, come ad esempio un promemoria per lanciare un’automazione o la comunicazione dell’esito di un’elaborazione)

L’unione di tutte queste funzionalità, relativamente molto semplici, permette di avere pieno controllo del mondo RPA, sia da altri applicativi aziendali che da dentro lo stesso gestionale.

Per rendere il prodotto ancora più potente, si possono integrare funzionalità di interazione con specifici framework proprietari. Ad esempio l'alimentazione automatica di code di lavorazione all'interno di un orchestratore, in base al punto in cui è arrivato il processo. Oppure l'inserimento/cancellazione automatica di permessi legati alle singole automazioni, all'interno del sistema di sicurezza del framework RPA.

Conclusioni

Il fiorire e il diffondersi della RPA apparentemente avvalora la tesi di chi la ritiene uno strumento proficuo a prescindere da come lo si usi e che in tempi celeri e con pochi sforzi riesca a cancellare grosse moli di debiti tecnici. Ma si fa presto a disilludersi se nelle aziende che l'adottano ne viene fatto abuso (in primis la non temporaneità di utilizzo) e se prevalgono le automazioni che "vengono funzionate" (manualismi) rispetto a quelle che funzionano bene da sé. Inoltre alcune volte, con grandi sforzi, si riescono a risolvere una miriade di piccole lacune IT che, sommate tra di loro, potrebbero corrispondere all'entità solo di una piccola parte delle lacune più grosse.
Nella maggior parte degli scenari, però, la RPA riesce essere d'aiuto anche quando zoppica. Migliorando la qualità del lavoro di molti impiegati, diffonde positività e accresce la fiducia, anche nei confronti dei dirigenti che hanno avuto a cuore queste questioni. A volte può riuscire anche a smuovere le acque: ad esempio se un'azienda avesse dei comparti IT un po' arrugginiti, introdurre la RPA farebbe sembrare quest'ultima la cosa più efficiente ed efficace che si sia mai vista e ciò potrebbe motivare a un risveglio il resto dell'IT. Quindi la conclusione è duplice:

- da un lato la RPA non potrà mai rimpiazzare un vero investimento sull'IT tradizionale che, se ben fatto, è risaputo dare in poco tempo grandi frutti ed accrescere la produttività aziendale
- dall'altro la RPA, se affidata ad informatici ed analisti esperti in questo campo, può dare un buon aiuto a fare respirare molti settori aziendali e ad offrire temporaneamente copertura al resto dell'IT

Precisazioni, riconoscimenti e ringraziamenti

Questo articolo ha trattato un cospicuo numero di temi ed aspetti sui quali riflettere, che rappresentano il mio pensiero. Devo precisare però che i contenuti non sarebbero stati nemmeno la metà senza aver condiviso l'esperienza lavorativa con due persone, che ringrazio:

- Franco Di Mauro, perché quasi tutti i concetti chiave, di ordine generale, che ho espresso nella prima parte dell'articolo sono rielaborazioni schematiche delle sue idee e dissertazioni e della sua visione aperta e mai scontata su questo tema (come su altri, per la verità)
- Carlo Gherbi, che, oltre ad avere ideato i sistemi che ho citato per sfruttare gli EDM in ambito RPA, da responsabile team di Process Automation, ha creduto nelle mie proposte e mi ha permesso di sperimentare e saggiare l'efficacia di alcune delle soluzioni tecniche che ho descritto in questo articolo

Infine, ringrazio tutti coloro i quali via email vorranno segnalare obiezioni e correzioni costruttive su questo articolo o vorranno maggiori ragguagli (ammesso che io sia in grado di darli) su temi che non ho ben chiarito o approfondito od ho tralasciato.