blueprism

Software Robots - the Virtual Workforce

Blue Prism Learning

PROCESS SOLUTION DESIGN

VERSION: 1.0.7

For more information please contact:

info@blueprism.com | UK: +44 (0) 870 879 3000 | US: +1 888 757 7476

www.blueprism.com

Contents

Blue Prism Limited, Centrix House, Crow Lane East, Newton-le-Willows, WA12 9UY, United Kingdom
Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: www.blueprism.com

1. Introduction

Blue Prism 은 자동화된 프로세스를 생성하여 안전한 가상 환경에서 실행하기 위한 플랫폼입니다. 자동화된 프로세스는 무인 및 관제 없이 실행될 수 있어야 하며, 이 문서는 이 원칙을 솔루션 설계의 중심으로 만드는 방법을 설명합니다.

2. Designing for Unattended Automation

블루 프리즘 프로세스는 자동으로 실행되는 일련의 복수의 수동 작업으로 구성됩니다. 무인 작업을 수행하고 예기치 않은 상황에서 복구하고 여러 시스템에서 동시에 작업할 수 있을 정도로 강력해야 합니다. Blue Prism 솔루션 설계를 배우는 때에 이러한 실현이 종종 누락됩니다.

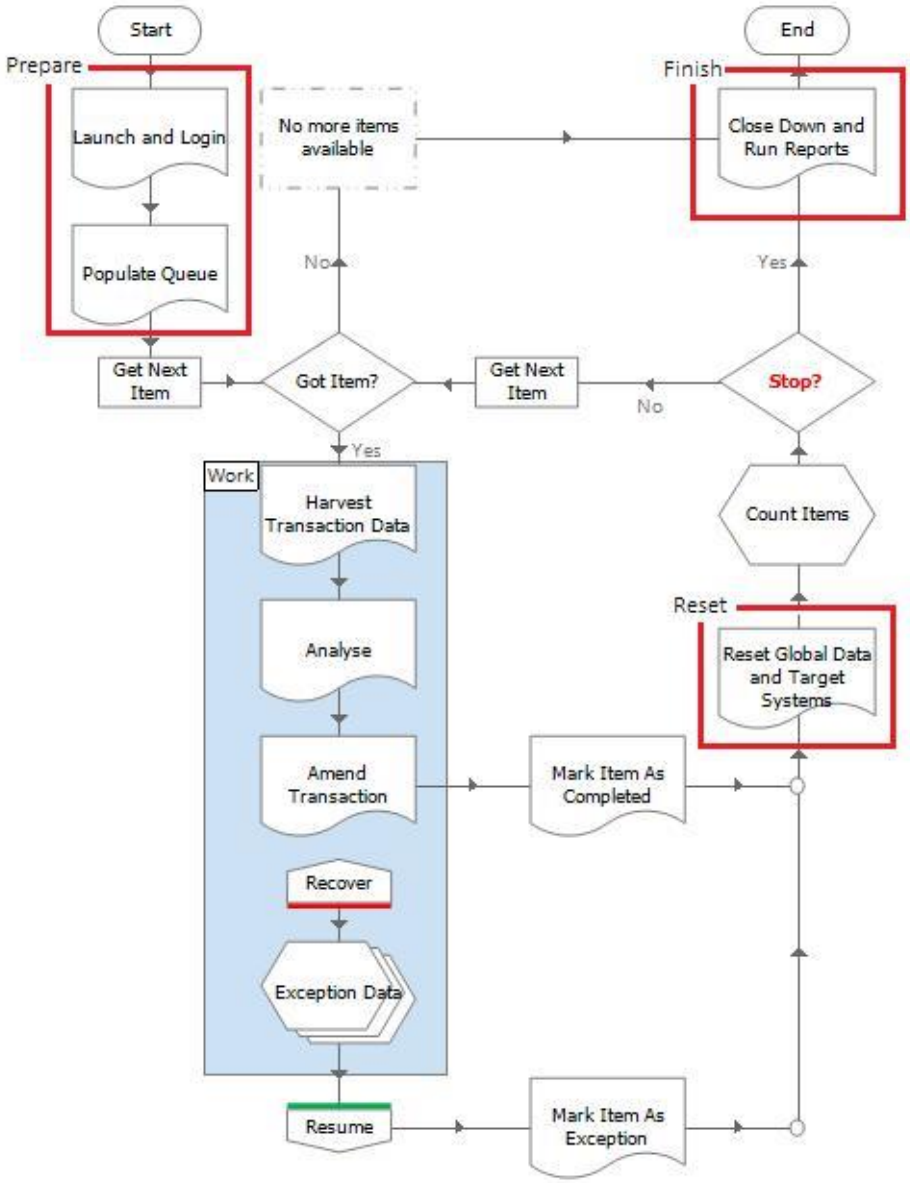
PDD 에서 수동 프로세스의 정의에 중점을 둔 후, 자동화된 프로세스를 단순히 일련의 작업 단계로 생각하는 것이 일반적인 실수입니다.

프로세스의 목표는 한 번에 하나씩 작업하는 것이기 때문에 너무 단순하며 한 번의 작업을 마친 후에는 다음 위치에 대비하여 시작 위치로 돌아와야 합니다. 따라서 프로세스는 사용중인 응용 프로그램 및 데이터에 대한 제어 권한을 유지해야 하며, 이러한 재설정 단계에는 기본 메뉴 화면으로 돌아가서 변수를 재설정해야 할 수도 있습니다.

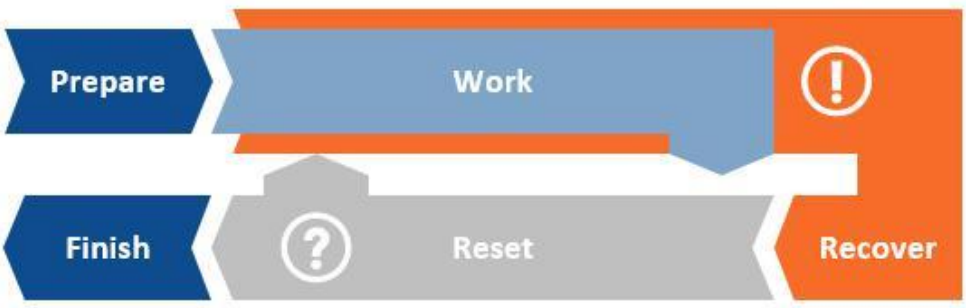
어떤 시점에서 프로세스는 이 루프에서 벗어날 필요가 있으며, 이는 종종 더 이상 처리할 문건이 남아 있지 않을 때이지만, 업무 일이 끝날 때와 같은 다른 이유로 결정을 내릴 수도 있습니다.

또한 작업 루프 전후에 추가 단계가 있을 수 있습니다. 준비 단계의 예는 응용 프로그램에 로그인하고 입력 데이터를 수집하는 것과 같습니다. 마무리 단계는 애플리케이션 로그 아웃 및 종료, 보고서 작성 또는 이메일 알림 전송과 같은 것일 수 있습니다.

이를 설명하기 위해 Blue Prism Basic Template 을 사용하여 구성된 다음 프로세스를 고려하십시오. 작업 처리(work) 외부의 세 영역이 명확하게 표시됩니다.



프로세스를 설계할 때의 일반적으로 간과되는 또 다른 점은 문제가 발생하지 않으며 항상 '행복한 길'을 유지한다고 가정하는 것입니다. 다시 말하면, 이것은 비현실적이며 아래에 주황색으로 표시된 것처럼 '행복하지 않은 길'의 논리를 충족시키기 위해 복구 단계를 포함해야 합니다.



여기서 예외 처리는 프로세스가 예기치 않은 애플리케이션 동작에서 복구되도록 하기 위해 수행되므로 케이스를 수동 조사를 위해 따로 보관하거나, 필요한 경우 프로세스에서 재작업할 수 있습니다. 복구 논리는 프로세스가 계속 작동하기 위해 응용 프로그램을 다시 시작하는 기능이 필요할 수도 있습니다.

3. Solution Types

수동 프로세스에 적용할 수 있는 다양한 유형의 자동화 솔루션이 있습니다. 수동 단계 1-9 로 구성된 다음의 추상 프로세스를 고려하십시오.

3.1. Full Automation

이상적인 시나리오는 모든 수동 단계를 자동화하고 전체 수동 프로세스를 교체할 수 있는 경우입니다. 단순화를 위해 예외 사례를 수동으로 해결하려는 노력은 여기에 설명되어 있지 않습니다.

3.2. Partial Automation

완전한 자동화를 달성할 수 없는 경우 원래 수동 프로세스 내에서 일련의 연속 단계를 자동화할 수 있습니다. 나머지 수동 단계의 노력은 자동화된 단계로 인한 비용 절감으로 상쇄됩니다.

이러한 종류의 부분 자동화에서는 종종 자동화를 활성화하기 위해 나머지 수동 단계를 조정해야 한다. 예를 들어, 단계 M1 에서, 사용자는 단계 A2 가 소비하기 위해 구조화된 데이터를 준비할 필요가 있을 수 있다. 유사하게, 단계 M9 에서, 사용자는 A8 로부터 이관 자료의 수신자가 될 필요가 있을 수 있다.

3.3. Fragmented Partial Automation

연속된 자동화를 달성할 수 없는 경우, 자동화를 여러 단계 일련 작업으로 적용할 수 있습니다.

여기에서 전체 이점을 평가할 때 수동 단계와 자동 단계 사이의 연결을 고려해야 합니다. 이관 작업에 새로운 수동 노력이 필요한 경우, 자동화 아이디어를 포기할 수도 있지만, 동기화된 상호작용을 쉽게 달성할 수 있으면 조각났더라도 자동화가 작동할 수 있습니다.

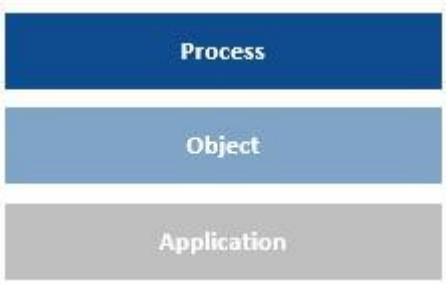
3.4. Restructured Partial Automation

파편화를 피하기 위해 연속적인 자동화 순서를 용이하게 하기 위해 일부 수동 단계를 리엔지니어링하거나 재정렬할 수도 있습니다.

여기에서 단계 1, 4 및 5 는 2, 3, 6, 7 및 8 을 하나의 직렬 프로세스로 자동화할 수 있도록 수동 순서로 배열되었습니다.

4. Solution Layers

Blue Prism 자동화 솔루션은 프로세스 계층에서 시작하여 오브젝트 계층 및 애플리케이션 모델, 애플리케이션 계층으로 통하는, 계층으로 인식될 수 있습니다.



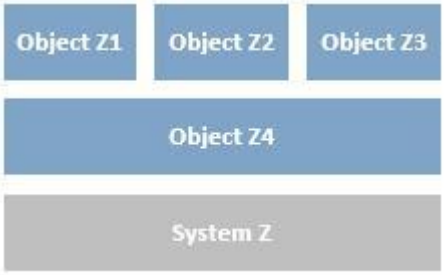
프로세스 계층과 오브젝트 계층의 역할은 처음에는 혼동될 수 있지만 본질적으로 오브젝트는 프로세스가 응용 프로그램을 조작하는 메커니즘을 제공하는 도구로 간주해야 합니다. 비즈니스 로직, 규칙 및 의사결정은 일반적으로 프로세스에 있어야 합니다.

4.1. Objects

비즈니스 객체는 프로세스가 응용 프로그램을 제어하는 데 사용하는 도구입니다. 이상적으로 객체는 프로세스가 복잡한 시퀀스를 오케스트레이션할 수 있는 간단한 함수 세트를 제공해야 합니다. 비즈니스 규칙 및 의사 결정에 대한 책임 객체를 전제로 다른 프로세스에서 객체를 재사용하고 객체 '라이브러리'를 구축할 수 있도록 하는 것이 목표입니다. 그리고 객체 라이브러리가 다양하게 증가함에 따라, 자동화된 솔루션을 제공할 때 드는 수고가 줄어들 것입니다.

자동화된 솔루션의 객체 계층은 하나 이상의 객체로 구성될 수 있으며, 종종 동일한 응용 프로그램의 여러 측면을 자동화하는 데 사용되는 여러 객체가 있습니다. 모든 응용 프로그램과 관련이 없는 일반 기능을 제공하는 데 사용되는 유틸리티 개체도 포함될 수 있습니다.

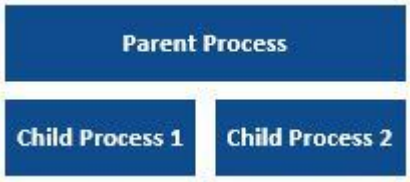
객체는 다른 객체를 사용할 수도 있으며 Surface Automation 과 같은 일부 상황에서는 다른 객체가 사용하는 대상 시스템과 상호 작용할 '기본' 객체를 생성하는 것이 바람직할 수 있습니다.



모든 객체가 대상 시스템과 상호 작용할 필요는 없습니다. 예를 들어 여러 프로세스에서 활용할 수 있는 규칙 엔진이 필요할 수 있습니다. 이것은 시작 파라미터를 통해 데이터를 받아들인 다음 출력 파라미터를 통해 결정을 다시 전달하기 전에 데이터를 조사하는 조치가 있는 오브젝트에서 개발될 수 있습니다.

4.2. Sub-processes and Wrapper Objects

프로세스는 다른 프로세스를 자식으로 호출하여 프로세스 계층을 만들 수도 있습니다.

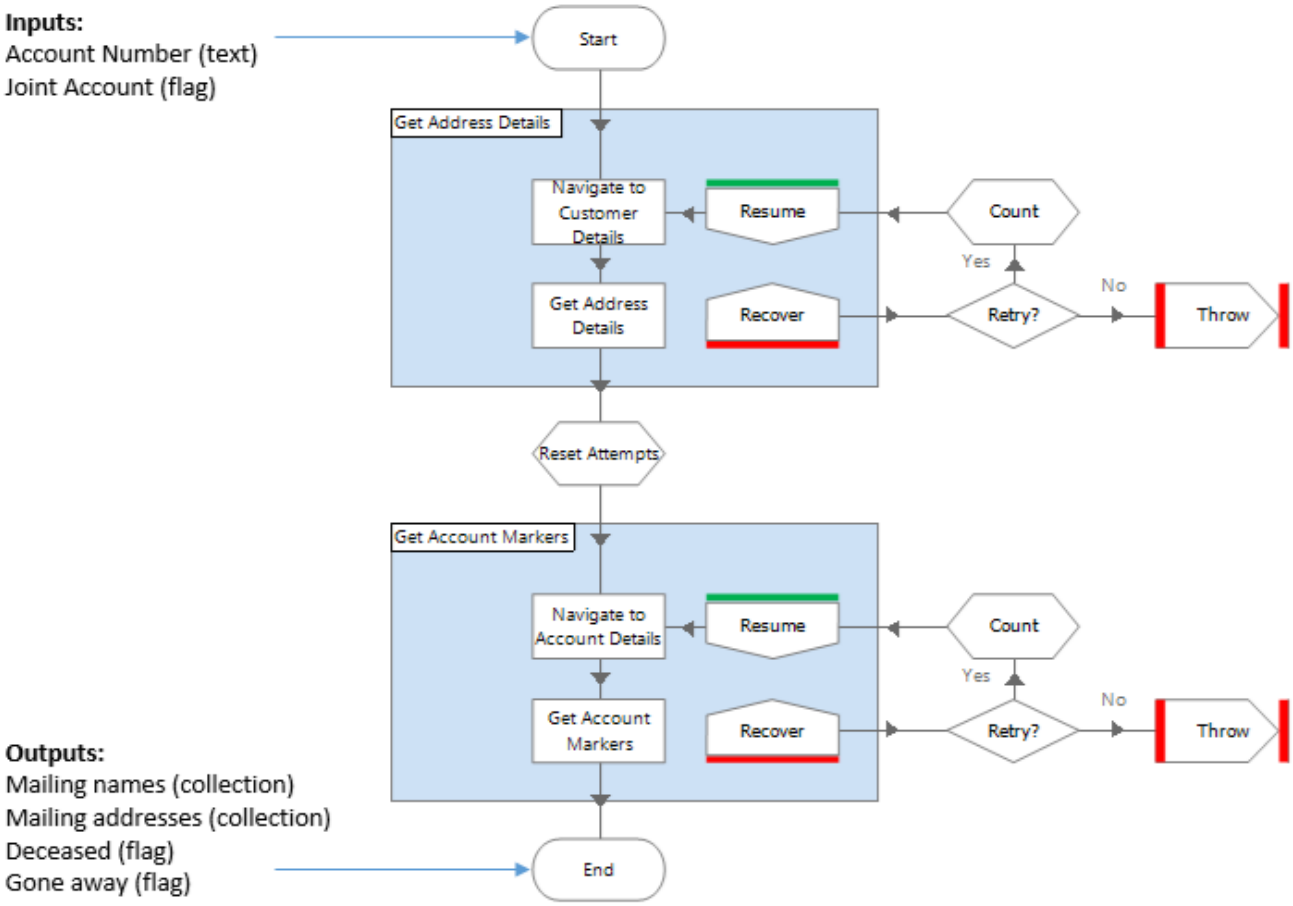


그러나 Blue Prism 이 메모리를 관리하는 방식은 자식 프로세스 사용에 대해 신중하게 고려해야 합니다. 객체와 달리 자식 프로세스는 부모에 의해 메모리에 보존되지 않습니다. 하위 프로세스가 종료되면 상위 프로세스는 .Net 가비지 콜렉터가 회수할 메모리를 해제합니다. 이로 인해 하위 프로세스가 상위 프로세스에서 반복적으로 사용되는 경우 가비지 콜렉터가 정리할 수 있는 것보다 하위 프로세스 (및 해당 오브젝트)가 메모리를 빠르게 사용하는 경우 원하지 않는 메모리 누수가 발생할 수 있습니다.

따라서 반복적으로 호출되는 하위 프로세스를 사용하지 않는 것이 좋습니다 (예: 반복 작업을 하는 경우. 다른 방법으로 객체를 사용하여 다른 객체의 동작을 래핑합니다.

예를 들어, 편지를 보내기 전에 고객 우편 주소 세부 정보를 수집할 때 공통 단계가 있는 경우를 예로 들 수 있습니다. 고객 서신을 주문하는 각 프로세스에서 이를 복제하는 대신 모든 프로세스를 호출할 수 있는 독립형 오브젝트로 모든 조치를 랩핑할 수 있습니다.

아래 작업은 대상 시스템과 인터페이스하지 않은 객체에서 수행한 것입니다. 그러나 세 가지 다른 객체에서 네 가지 동작을 래핑합니다.



5. Design Basics

5.1. Recoverability

간단히 말하면, 복구 가능성은 문제를 처리하고 정상으로 돌아가는 솔루션의 역량입니다. 행복한 길이 유일한 가능성이라고 가정하는 것은 순진한 일이며 예상치 못한 상황에서 복구를 시도하기 위해 항상 준비해야 합니다.

System Recovery

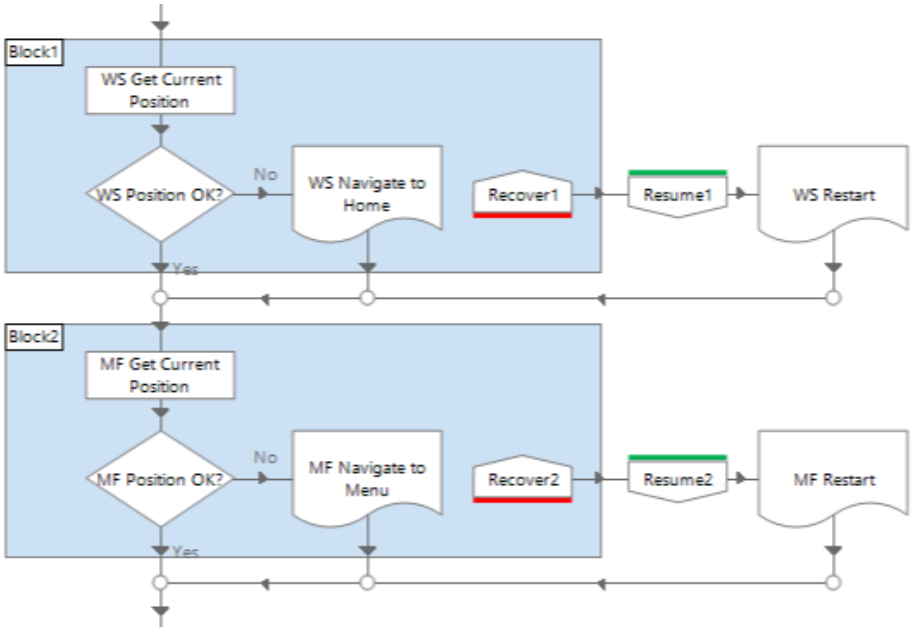
자동화된 솔루션이 항상 애플리케이션에 대한 제어를 유지하는 것이 중요합니다. 모든 것이 잘 진행되면 이는 비교적 간단하며, 확인해야 할 단 한가지는 다음 건을 작업하기 전에 응용 프로그램이 시작 위치로 돌아온다는 점입니다.

응용 프로그램이 예상대로 작동하지 않고 제어가 실종되면 어려움이 발생합니다. 일반적으로 이는 객체에서 '시간 초과' 또는 '요소를 찾지 못했습니다' 오류로 나타납니다. 그로 인한 예외는 예외 처리로 관리할 수 있지만, 프로세스를 계속하려면 응용 프로그램을 계속 제어 상태로 만들어야 합니다.

일부 응용 프로그램에서는 이 작업이 쉬울 수 있습니다. 예를 들어 웹 응용 프로그램에서는 홈 URL 로 직접 이동하면 되는 경우일 수 있으며, 메인 프레임에서는 모든 위치에서 되돌리는 범용 키 입력이 있을 수 있습니다. 하지만 때로는, 탐색하는 것이 그렇게 간단하지 않으므로, 응용 프로그램을 다시 시작해야 할 수도 있습니다. 다시 말하지만, 이는 고통스럽지 않으며 그저 애플리케이션을 종료하고 다시 시작하면 될 수 있습니다. 그러나 일부 응용 프로그램은 대충적인 처리를 원하지 않으며 규정된 로그 아웃 절차를 엄격하게 준수해야 합니다.

솔루션에 복구용 섹션을 포함할 것을 예상하여 각 애플리케이션을 신중하게 평가해야 합니다. 또한 솔루션 설계 문서 또는 프로세스 설계 지침에서 시스템 복구가 명시적으로 포함되지는 않아도, 개발자는 강력한 솔루션을 구성할 때 이 개념이 필요함을 이해해야 합니다.

다음은 다음 처리를 수행하기 전에 두 응용 프로그램(웹 시스템 및 메인 프레임)을 준비하는데 사용되는 복구 논리의 예입니다. 응용 프로그램의 현재 위치가 결정되며, 예상과 다르거나 시도가 실패하면 논리가 복구를 시도합니다. '탐색' 페이지는 예외를 발생시키고 다시 시작하기 전에 여러 번 시도합니다. '다시 시작' 페이지도 여러 번 시도하지만, 예외를 발생시키기 위해 예외를 처리하지 않은 채로 남겨 둡니다. 바람직하지는 않지만 제어가 손실되었고 처리가 방치된 채로 계속될 수 없음을 나타내기 위해 여기에 종료가 필요합니다.



Case Recovery

사례 복구는 프로세스를 다시 시작한 후 사례를 해결하는 기능입니다. 예를 들어, 프로세스가 케이스 처리 중에 있는 동안 외부 문제로 인해 Blue Prism 이 완전히 실패하면 해당 케이스는 어떻게 될까요? 수동 팀에게 예외로 넘겨 질 것이라고 간단히 말하는 것으로 충분할까요? 또는 프로세스가 이전 위치에서 사례를 계속 진행할 수 있어야 할까요? 그렇다면 어떻게 달성할 수 있을까요?

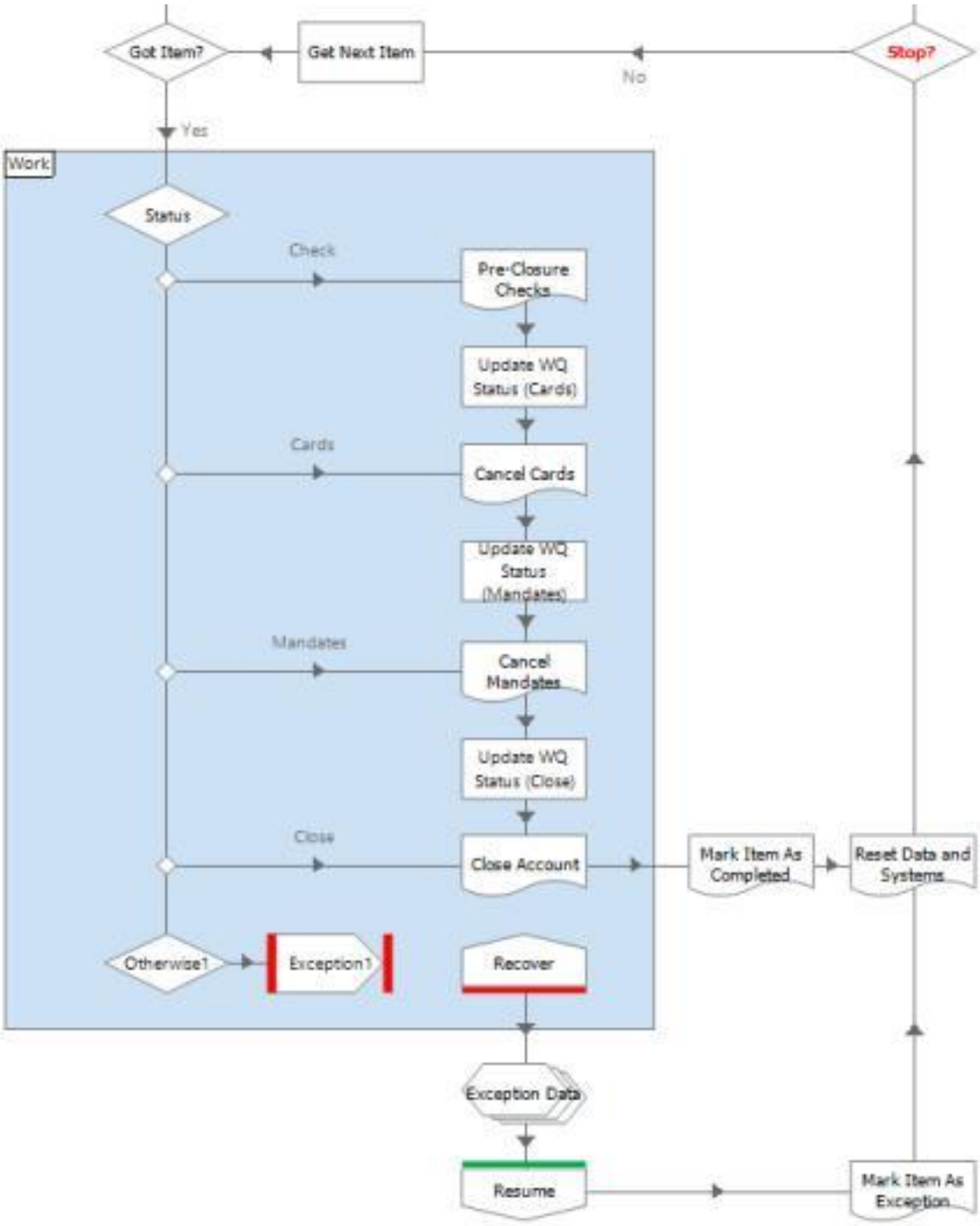
사례가 진행됨에 따라 대기열 항목을 업데이트하여 사례의 현재 상태를 기록할 수 있습니다. 그런 다음 갑작스런 충격이 발생한 경우라도, Blue Prism 프로세스 또는 수동 팀에 의해 사건이 다시 발생할 때, 이전 단계의 진행 상황을 알 수 있습니다.

아래 초록에서 3 단계 이후에 문제가 발생할 때까지 각 단계에서 사례의 상태가 기록됩니다.

사례가 재 작업되면 4 단계에서 시작할 수 있습니다.

이를 설명하기 위해 다음 예를 고려하십시오. 계정 폐쇄 작업 대기열은 사례 당 최대 3 번의 시도를 허용하도록 구성되었습니다. Blue Prism 은이 속성을 사용하여 예외로 표시될 때 대기열 항목을 복제하여 보류중인 항목을 새로 만듭니다.

아래의 계정 폐쇄 프로세스는 대기열 항목 상태 속성을 사용하여 사례 진행 상황을 추적하도록 설계되었습니다. 이 과정에는 Check, Cards, Mandates 및 Close 의 네 단계가 있으며 각 성공적인 단계 후에 상태가 업데이트됩니다. 이를 통해 프로세스에 제시된 재시도 사례가 올바른 다음 단계로 이동합니다. 또한 수동 검토를 위해 예외가 전송되면 상태 값은 사례가 실패했을 때 수동 팀에 표시됩니다.



Database Recovery

프로덕션 데이터베이스를 복구하는 방법을 솔루션 디자인의 일부로 고려해야 합니다. 프로덕션 데이터베이스가 실시간으로 복제 또는 미러링되지 않는 경우 재해로부터 복구하기 위해 백업을 복원해야 할 수 있습니다. 이러한 상황이 발생하면 백업을 수행한 후 처리된 사례가 대기열에서 미처리된 것으로 나타날 수 있으며 이러한 사례를 재 작업하려는 솔루션의 효과를 고려해야 합니다.

1000 건의 사례가 작업 대기열에 로드되고 처리 도중에 심각한 데이터베이스 장애가 있는 예를 고려하십시오. 백업 서비스는 4 시간 마다 증분 백업을 제공하지만 최신 백업이 복원되면 모든 사례가 보류 중으로 표시됩니다. 단순히 프로세스를 다시 시작하면 이미 작동한 500 개의 사례가 재 작업됩니다.

오래된 프로덕션 데이터베이스 복원 효과를 완화하는 것은 디자인 단계에서 가장 잘 수행됩니다. 프로세스에 따라 대상 응용 프로그램에 자연 방어가 있을 수 있습니다. 예를 들어 고객 계약을 취소하는 프로세스에서 대상 응용 프로그램이 이미 취소된 계약을 다시 취소할 수 없는 경우 해당 사례를 완료된 것으로 표시하도록 프로세스를 설계할 수 있습니다.

그러나 자금을 이체하는 재무 프로세스의 예에서 재 처리 사례는 재난으로 중복 지불을 할 수 있습니다. 그러나 프로세스가 중요한 단계에서 계정에 인식 가능한 메모를 적용하여 '발자국'을 찾아서 만들도록 설계된 경우 발자국이 있는지 확인하고 중요한 단계를 반복하지 않을 수 있습니다.

5.2. Scalability

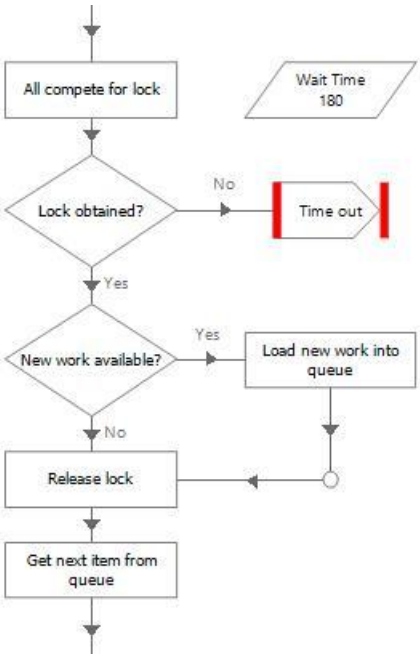
자동화된 프로세스를 설계할 때 일반적인 감독은 프로세스가 여러 시스템에서 동시에 실행되어야 한다는 것을 잊어버리는 것입니다. Blue Prism 작업 큐가 여러 프로세스 인스턴스가 동일한 큐 항목에 액세스하는 것을 방지하지만 프로세스가 둘 이상의 머신에서 실행되는 경우 케이스 작업 루프 외부 단계에 특별한 주의가 필요할 수 있습니다.

예를 들어, 매일 입력 파일을 소비하고 작업 큐를 로드하여 프로세스를 시작하는 일반적인 시나리오를 생각해 보십시오. 프로세스의 세 인스턴스가 실행중인 경우 어떻게 될까요? 그들은 모두 파일을 읽고 큐를 세 번 로드하려고 시도합니까? 또는 하루가 끝날 때 보고서를 작성한다고 가정하십시오. 중복 보고서가 작성되지 않도록 하려면 어떻게 해야 할까요?

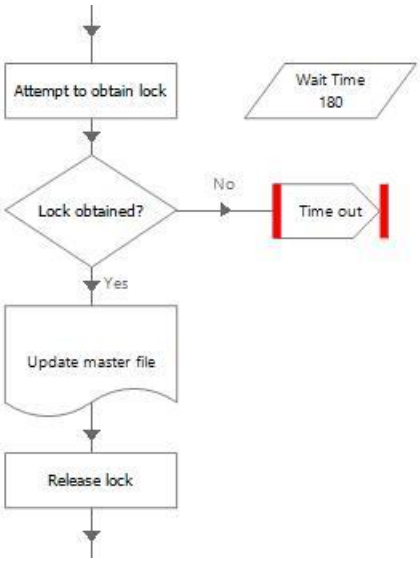
Environment Locks

환경 잠금은 프로세스와 객체 간에 '권한'을 공유할 수 있는 Blue Prism 기능입니다.

프로세스 인스턴스는 하나의 인스턴스만 특정 단계를 수행할 수 있도록 권한에 대해 경쟁하도록 만들 수 있습니다. 예를 들어 프로세스의 첫 번째 단계는 데이터 소스를 읽고 작업 큐를 로드하는 것입니다. 프로세스의 세 가지 인스턴스가 동시에 시작되고 모두 환경 잠금 보유를 위해 경쟁합니다. 승자는 하나만 있을 수 있으며 해당 인스턴스는 대기열을 로드하고 패자는 대기합니다. 잠금이 해제되면 패자는 다시 경쟁합니다.



다른 요구 사항은 특정 단계를 동시에 수행할 수 있는 인스턴스 수를 제어하는 것입니다. 예를 들어, 둘 이상의 사용자가 열 수 없는 파일을 업데이트하는 프로세스를 상상해보십시오. 환경 잠금은 파일에 대한 '한 번에 하나' 액세스를 보장하고 프로세스 인스턴스가 파일을 업데이트 할 차례를 기다리도록 하는데 사용됩니다.



5.3. Reusability

재사용성은 Blue Prism RPA 전달 방법론의 핵심 원칙입니다. 오브젝트, 랩퍼 오브젝트 및 서브 프로세스를 신중하게 설계함으로써 재사용 가능한 로직 라이브러리를 구축하고 제공 노력을 줄이고 유지 보수 오버 헤드를 최소화 할 수 있습니다.

객체에는 '비즈니스 프로세스 로직'이 없어야 하며 객체 페이지는 작고 기계적인 단계를 수행해야 합니다. 개체의 기능은 응용 프로그램을 조작할 수 있는 도구를 프로세스에 제공하는 것이며, 개체 논리가 일반적 일수록 다른 프로세스에서 재사용할 가능성이 높습니다.

예를 들어 비즈니스 프로세스의 첫 번째 단계가 'Open MediSys 및 환자 세부 정보 가져 오기'인 경우 첫 번째 직관은 이를 수행하는 개체 페이지를 만드는 것입니다. 그러나 다른 프로세스가 'Open MediSys 를 열고 임상의 진료 예약을 받으십시오'로 시작하면 기존 MediSys 로직을 개체를 변경(및 테스트)하지 않고 재사용할 수 없습니다.

'시작', '로그인', '환자 찾기' 및 '환자 세부 정보 읽기'와 같이 기본 단계를 수행하는 일련의 페이지를 디자인하는 것이 더 좋습니다. 이렇게 하면 두 번째 프로세스는 첫 번째 프로세스에 대해 생성된 일부 논리(예 : '시작'및 '로그인')를 활용할 수 있습니다. 따라서 개체 기능의 세분성을 높이고 응용 프로그램의 한 화면에서 '원자' 작업(예 : 읽기, 쓰기 또는 탐색)을 수행하는 페이지를 사용하면 개체를 더 재사용할 수 있습니다.

일반적으로 큰 객체도 피해야 합니다. 페이지 수가 많은 개체는 작동하지만 비 효율성을 가져올 수 있습니다. 네트워크를 통해 큰 객체를 전송하려면 더 많은 네트워크 대역폭이 필요합니다. 사용자는 얼마나 사소하게 사용하든 관계없이 전체 객체를 PC 메모리에 커밋합니다. 한 번에 한 명의 개발자만 객체를 작업할 수 있습니다. 단일 객체에 대한 의존성이 증가함에 따라, 다중 프로세스에 영향을 미치는 객체 결함의 영향이 증가합니다.

이러한 문제를 피하려면 응용 프로그램 통합 논리를 하나의 큰 객체로 집중시키지 않고 일련의 작은 객체로 분리하는 것이 좋습니다. 이 방법은 여러 가지 장점을 제공합니다. 경량 객체는 적은 대역폭, 메모리 및 디스크 공간을 소비합니다. 객체 그룹은 팀이 응용 프로그램 통합을 보다 쉽게 개발할 수 있도록 합니다. 손상된 물체의 잠재적인 영향이 최소화됩니다.

자세한 내용의 샘플 객체 설계 지침을 찾으려면 다음을 검토하십시오:

- Solution design example documents
- Blue Prism Delivery Roadmap within Lifecycle Orientation

6. Case Management

6.1. Reset

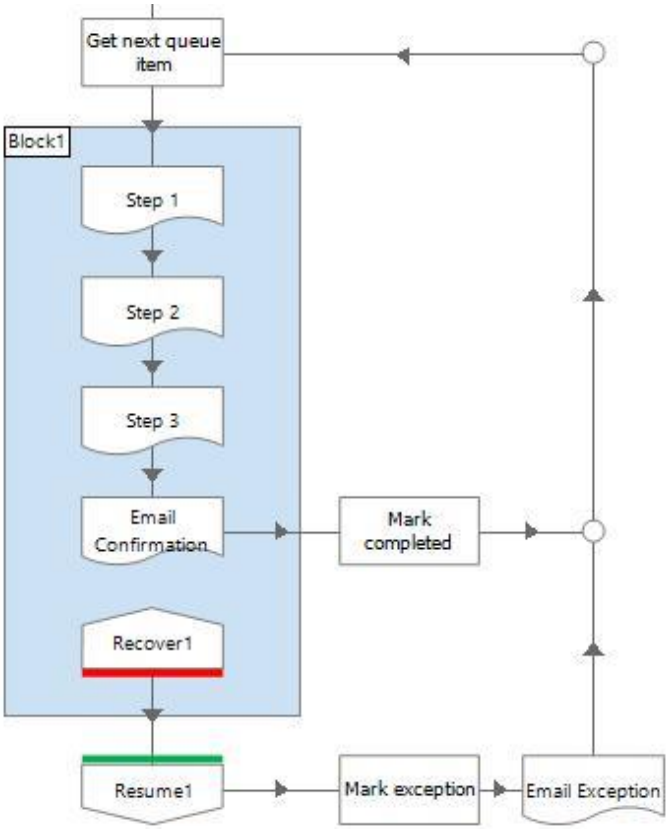
거의 모든 프로세스에는 하나의 사례가 차례로 처리되는 기본 루프가 포함됩니다. 데이터 항목은 재사용되므로 이전 사례의 값이 실수로 다음 사례에 사용되지 않도록 주의해야 합니다.

위에서 언급했듯이 프로세스는 수명 기간 동안 객체를 메모리에 유지하므로 객체 계층 내의 데이터 값이 유지될 수 있습니다. 데이터가 대/소문자를 구분하지 않는 경우 바람직하지만 현재 사례에 영향을 주는 오래된 사례 데이터를 피하는 것이 중요합니다.

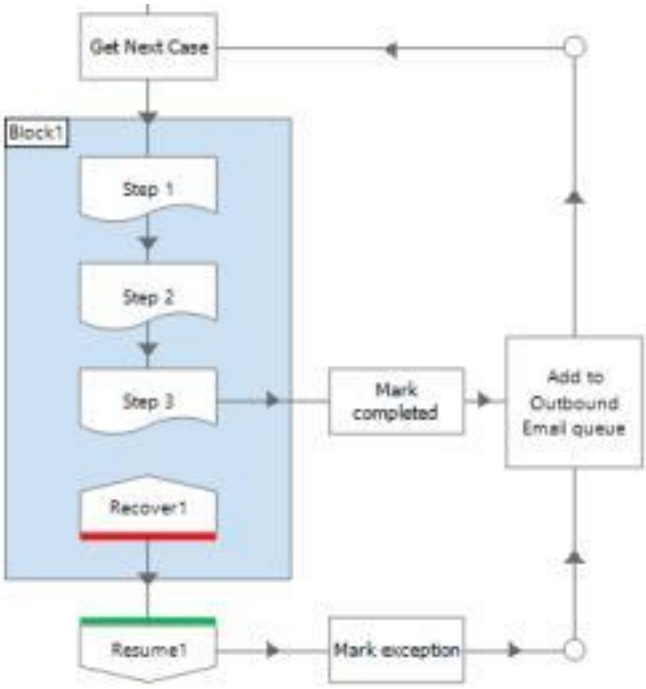
6.2. Resilience

다시 시작한 후 사례를 복구하는 기능과 마찬가지로 솔루션 디자인은 복원력을 향상시키고 예외 사례 수를 줄일 수 있습니다. 사례를 일련의 작업으로 세분화하는 데 여러 대기열을 사용할 수 있습니다. 예를 들어, 이메일 또는 웹 서비스를 통해 각 사례가 끝날 때마다 결과를 전달해야 하는 경우 이 작업을 보조 대기열에서 별도의 사례로 고려하는 것이 좋습니다. 이렇게 하면 필요한 경우 나중에 작업을 완료하거나 다른 컴퓨터에 위임할 수 있습니다.

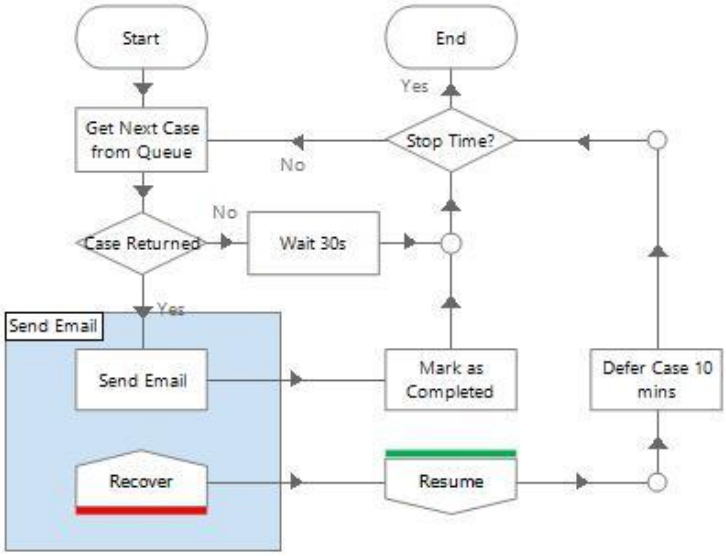
이 (고의적으로 단순화 된) 예에서 프로세스는 사례를 처리하고 결과를 전자 메일로 보냅니다. 예를 들어 메일 서버를 사용할 수 없어서 확인 이메일을 보낼 때 예외가 발생하면 해당 사례는 예외로 표시되고 수동 검토를 위해 전송됩니다. 사건이 본질적으로 완료되었으므로 이것은 불필요합니다.



그러나 아래의 예에서 프로세스는 이메일을 보내지 않고 대신 다른 프로세스가 보낼 작업 큐에 메시지를 추가합니다.



별도의 '이메일 보내기' 프로세스는 다른 프로세스와 동시에 작동합니다. 이 프로세스는 대기열에 항목을 추가하는 다른 프로세스 대신 이메일을 보낼 수 있습니다. 이메일 전송에 실패한 경우 사례는 단순히 연기되어 나중에 다시 시도됩니다. 이메일 게이트웨이에 중대한 문제가 있는 경우 일단 수정되면 이메일이 발송됩니다.



7. Workload Management

7.1. Accountability

거의 모든 자동화된 솔루션은 파일 읽기, 받은 편지함 폴링, 웹 서비스 폴링 등 어떤 종류의 워크로드를 소비합니다. 불가피하게 이 데이터는 감사 및 MI 를 위해 항상 Blue Prism 작업 대기열에 로드 될 뿐만 아니라 프로세스의 여러 인스턴스 간에 작업을 안전하게 배포하는 수단으로도 사용됩니다.

이를 위해 모든 작업을 설명할 수 있는 솔루션을 설계하고, 사례를 잃어 버리거나, 중복되지 않으며, 모든 결과를 입증할 수 있는 솔루션을 설계해야 합니다. 구현된 솔루션은 보다 광범위한 운영의 일부가 되므로 제공된 작업을 안정적으로 소비할 뿐만 아니라 신뢰할 수 있는 결과를 비즈니스에 다시 보고하도록 설계해야 합니다. 최소한 이 의사 소통은 예외 사례와 위탁을 위한 것이지만 완료된 사례에 대한 세부 정보도 포함해야 할 수도 있습니다.

필요한 책임 수준을 달성하려면 각 사례를 처리하는 동안 수집된 입력 데이터 정보를 업데이트해야 할 수 있습니다. 이는 대기열 항목 상태 필드를 설정하거나 태그를 적용하는 것처럼 간단할 수 있지만 대기열 항목 데이터 콜렉션에 데이터를 쓰는 것을 포함할 수도 있습니다.

7.2. Balance

워크플로 도구 또는 데이터베이스와 같은 다른 응용 프로그램에서 작업을 수행하기 위해 자동화된 솔루션이 필요한 경우는 드문 일이 아닙니다. 발생점으로 돌아가서 수작업 결과로 업데이트해야 할 수도 있습니다. 데이터 소스와 자동화된 솔루션의 균형을 유지하고 케이스가 작동하지 않고 중복도 발생하지 않으며, 결과 보고가 누락되지 않도록 설계자가 주의를 기울여야 합니다.

일부 설계에서는 여러 개의 Blue Prism 작업 대기열을 사용해야 할 수도 있으므로 평형이 유지되도록 반복해서 숙고되어야 합니다.

7.3. Overload

작업 부하가 급증하거나 SLA 가 엄격할 수 있는 경우 임박한 문제를 감지하고 알리는 메커니즘을 설계하는 것이 정당화될 수 있습니다. 예를 들어, 보류중인 항목 수에 평균 사례 시간을 곱하면 완료 시간 예측을 SLA 와 비교할 수 있습니다.

마찬가지로, 비정상적으로 작거나 존재하지 않는 워크로드도 자동화 된 경고를 보증하거나 대체 조치를 트리거 할 수 있습니다.

8. Data Management

8.1. Preservation

기능적 요구 사항은 솔루션 데이터가 보존되는 기간을 나타내야 하지만, 어떤 종류의 제어가 구현되지 않는 한 일부 유형의 데이터는 무기한으로 지속될 것임을 인식하는 것이 중요합니다. System Manager 를 사용하여 세션 로그를 보관할 수 있으며 로그 데이터의 증가를 제한하기 위해 보관 정책을 마련해야합니다.

그러나 자동화된 메커니즘이 작성되지 않으면 작업 큐 데이터는 데이터베이스에 남아 있습니다. 이는 단일 '대기열 관리' 프로세스의 형태이거나 각 프로세스가 자체 대기열을 유지할 수 있도록 '내부 – 작업 큐' 객체의 기능을 사용할 수 있습니다. 일반적으로 '유지 관리'는 단순히 워크로드, MI 또는 감사의 일부로 더 이상 목적이 없는 오래된 항목을 삭제하는 경우입니다.

마찬가지로 모든 입력 및 출력 파일의 수명을 다루고 자동화된 제어 절차를 고려해야 합니다.

8.2. Security

일부 시나리오에서는 중요한 데이터를 처리해야 할 수 있으며 이는 솔루션 디자인에 영향을 줄 수 있습니다. 기본 레벨에서 작업 큐에서 암호화가 사용 가능해야 하며 제어실 또는 세션 로그에 키 데이터가 노출되지 않도록 주의해야 합니다. 더 중요한 것은 솔루션의 어느 곳에나 주요 데이터가 저장되지 않도록 규정하는 보안 정책이 있을 수 있습니다. 이러한 요구 사항으로 인해 데이터가 완전히 일시적이며 대기열, 로그 또는 다른 곳에 유지되지 않도록 솔루션을 설계해야 할 수 있습니다. 예를 들어, 신용 카드 번호는 '적시에' 가져와야 하며, 일단 사건이 발생한 후에도 흔적이 남지 않도록 각별히 주의해야 합니다.

9. Efficiency

9.1. Integration Efficiency

대체 응용 프로그램 통합 옵션이 있는 경우가 많으므로 다양한 기술의 장단점을 고려해야 합니다. 응용 프로그램에 사용자 인터페이스를 모델링하는 대신 사용할 수 있는 API 또는 웹 서비스가 있을 수 있습니다. 또는 사용자가 파일을 열지 않고도 파일을 직접 읽을 수 있습니다.

수동 단계를 충실히 재현하는 것이 자연스러운 경향이만 Blue Prism 이 사람이 이용할 수 없는 메커니즘을 이용할 수 있는 경우가 있을 수 있습니다. 기본적으로 로봇은 더 많은 정보를 메모리에 저장할 수 있으므로 사람이 두 애플리케이션에서 정보를 수집하기 위해 앞뒤로 이동해야 하는 경우 로봇은 필요한 모든 것을 한 번에 읽을 수 있습니다.

기본적으로 자동화된 솔루션은 '있는 그대로' 수동 프로세스를 따라야 하지만 로봇이 인간보다 능력이 뛰어난 경우 효율성을 높일 수 있는 범위가 있을 수 있습니다.

9.2. Exception Efficiency

자동화된 솔루션은 기술적인 이유로 또는 범위를 벗어난 이유로 자동으로 해결할 수 없는 사례를 예상해야 합니다. 설계는 이러한 경우의 수동 해결이 어떤 식으로든 도움을 줄 수 있는 지 여부를 고려해야 합니다. 예를 들어 대기열 항목 상태 필드를 사용하여 다른 예외 범주를 나타낼 수 있습니다. 또는 사례를 처리하는 동안 읽은 시스템 데이터를 정기적으로 대기열에 다시 저장한다면, 이 추가 정보를 예외 보고서에 포함시켜 수동 작업자가 시스템을 탐색하여 사례 데이터를 다시 수집할 필요가 없도록 도와줄 수 있습니다.

예외가 발생한 후 작업을 계속할 가치가 있는 상황이 있을 수도 있습니다(예 : 후속 수동 참조 작업을 최소화하기 위해 데이터를 가져 오거나 궁극적으로 예외로 표시하기 전에 사례를 계속 작업하는 경우).

부모 / 자식 (즉, 중첩된) 사례를 사용하는 경우 예외 후에 계속해서 하위 사례를 처리하는 것이 더 효율적일 수 있습니다. 그 이유는 즉시 멈추는 것보다 자식 중 90%의 예외를 부모의 예외로 표시하는 것이 좋습니다.

9.3. License Utilisation

여러 프로세스로 구성된 솔루션을 설계할 때 라이센스 활용도 고려해야합니다. 별도의 프로세스가 최적의 구성이 될 수 있지만 라이센스가 중요한 경우 단일 프로세스(아마도 하위 프로세스, 환경 잠금 또는 다중 큐 사용)에 대해 고려해야 합니다.

10. Notification

고객의 요구 사항에 따라 특정 이벤트에 대해 경고해야 하며, 이러한 알림을 언제, 어디서, 어떻게 발행할 것인 지를 설계해야 합니다.

예를 들어, 제어 팀은 프로세스 종료와 같은 주요 이벤트에 대한 알림을 받을 수 있습니다. 자동화가 완료된 위치에서 처리를 완료할 수 있도록 프로세스 완료를 오퍼레이션에 알려야 합니다. 또는 더 심각한 것은 비상 계획을 실행하기 위해 SLA 위반에 대해 경고해야 할 수도 있습니다.

이유가 무엇이든, 솔루션의 커뮤니케이션 방식은 디자인의 일부로 고려해야 합니다. 전자 메일이 일반적으로 선호되는 방법이지만 헬프 데스크에 대한 SNMP 메시지도 사용할 수 있습니다. 또는 워크플로 도구 또는 MI 데이터베이스와 같은 다른 응용 프로그램에 대한 업데이트로 알림을 발행할 수 있습니다.

11. Design Procedure

생산 환경에 Blue Prism 솔루션을 제공하는 것은 많은 IT 프로젝트에 익숙한 Define-Design-Build-Test-Implement 의 표준 경로를 따릅니다. Blue Prism 은 함께 제공되는 문서 템플릿과 함께 검증된 설계 방법론을 제공하며, 교육생은 Lifecycle Orientation 모듈의 일부로 이들을 소개합니다. 그러나 이러한 템플릿은 규범이 아니며 일부 클라이언트는 RPA 에 적합하도록 자체 방법론과 문서를 보다 편하게 적용할 수 있습니다. 중요한 점은 문서의 목적이며, 문서 자체는 단지 도구일 뿐입니다. 요구 사항을 정의하고 세부 사항을 노출하며 합의에 도달하기 위한 것입니다.

11.1. 'As is' Definition and Requirements

이 문서의 다른 곳에서 언급한 것처럼 프로세스 정의 문서(PDD)는 '있는 그대로' 수동 프로세스를 설명하는 데 사용됩니다. 이 정보는 이미 SOP(States of Procedure) 또는 기타 프로세스 문서에 있을 수 있지만, 생각하지 않은 로봇이 프로세스를 수행하도록 지시할 수 있을 정도로 충분히 상세해야 합니다.

수동 프로세스의 정의와 함께 자동화 솔루션에 대한 프로세스 소유자의 희망 사항을 문서화해야 합니다. FRQ(Functional Requirements Questionnaire)는 자동화된 솔루션의 작동 방식을 설명하는데 사용되는 것이 아니라 프로세스 소유자와 인터뷰하여 솔루션 운영 방식에 대한 상위 수준의 요구 사항을 추출하는 방법입니다. 예를 들어 월요일부터 금요일까지 업무 시간 동안 또는 다른 일정으로 실행해야 한다면 예외를 어떻게 전달해야 할까요?

11.2. 'To be' Design

PDD 및 FRQ 가 완료되면 설계자는 정보를 솔루션 디자인 문서(SDD)로 변환할 수 있습니다. '있는 그대로'와 '존재할 것'을 서로 다른 문서로 분리하면 오해와 누락을 방지할 수 있습니다. PDD 와 마찬가지로, 워크숍 시나리오에서 SDD 를 걷는 것은 제안을 설명할 뿐만 아니라 단점을 드러내는 훌륭한 방법이기도 합니다.

또한 SDD 는 개발 리드에게 제안된 솔루션의 품질을 평가하고 기존 로직(즉, 객체 라이브러리)의 충분히 사용되고 있으며 귀중한 프로젝트 시간이 낭비되지 않는 지 확인할 수 있는 기회를 제공합니다. 설계가 사인오프된 상태에서 PDI(Process Design Instruction) 및 ODI(Object Design Instruction)를 통해 개발을 시작할 수 있습니다. 이 두 문서는 개발자에게 빌드 방법과 내용을 알려주는 디자이너의 수단입니다.

11.3. Shock Proof Design

설계는 이론적인 '충격 테스트'로 테스트해야 합니다. 이는 네트워크 중단과 같이 솔루션이 갑작스러운 타격에 어떻게 대응할 것인지를 상상하는 것입니다.

프로세스가 케이스 처리 중에 있고 모든 네트워크 통신이 끊어지고 데이터베이스 연결이 끊어지고 프로세스가 종료된다고 가정하십시오. 네트워크가 복원되고 프로세스가 다시 시작되면 어떻게 됩니까? 이전 사례는 어떻게 됩니까? 프로세스가 재 작업하는 경우 단계를 복제하거나 중단된 부분을 선택합니까? 응용 프로그램이 열려 있으면 프로세스가이를 처리할 수 있습니까? 케이스 중간에 발생하는 갑작스런 이벤트뿐만 아니라, 큐를 로드하는 동안 이벤트가 발생하면 어찌 될까요, 충격이 사라지면 복제본이 추가되거나 케이스가 손실됩니까?

그러한 상상은 비관적으로 보일지 모르지만, 오작동 솔루션의 결과가 충분히 심각하면 매우 필요할 수 있습니다.

11.4. Application Assessment

새로운 대상 시스템이 범위에 들어올 때마다 항상 응용 프로그램 평가를 수행할 가치가 있습니다. 이를 위해서는 새로운 시스템의 요소에 대해 Application Modeller 스파이를 실행하여 식별할 수 있는 양과 방법을 쉽게 확인할 수 있습니다. 일반적으로 Windows, 브라우저 및 Java 애플리케이션의 경우 이 활동을 수행하려면 개발자가 각 요소 유형(콤보 상자, 확인란, 테이블, 목록 등)과 인터페이스하기 위한 가장 적합한 기술을 결정해야 합니다. 이를 통해 개발 노력을 추정하고 사례 처리 시간의 초기 근사치를 제공할 수 있습니다.

11.5. Project Types

Proof of Concept

POC 의 목적은 무인 자동화가 가능한 완전 자율 솔루션을 생산하기보다는 자동화의 잠재력을 입증하는 것이어야 합니다. 이를 위해 디자인은 확장성, 응용 프로그램 제어, 예외 처리, 알림 및 MI 와 같은 프로덕션 솔루션에 필요한 기능에 중점을 둔 범위 내 사례에 중점을 두어야 합니다.

분명히 솔루션이 작동해야 하지만 완전한 프로덕션 제품일 필요는 없으며 데모 목적으로 종종 참석 모드에서 제한된 기간 동안만 실행됩니다. 디자인은 가능한 시간에 현실적으로 제공할 수 있는 것을 고려하고 프로젝트의 전반적인 목적이 무엇인지 기억해야 합니다. 개념에 대한 증명 후에 설계 검토가 이루어져야 하며, 최종 설계 문서화 및 추가 개발이 통합되어 프로세스가 생산 강도가 높고 공식 테스트 준비가 되었는 지 확인해야 합니다.

Pilot

파일럿은 프로덕션 사례에 대해 일시적으로 실행되도록 구축된 프로세스입니다. POC 와 유사하게 파일럿 프로젝트의 설계는 프로젝트의 목표를 반영해야 합니다. 파일럿의 목적은 본격적인 솔루션을 생성하는 것이 아니므로 솔루션에서 수행할 것과 수행하지 않을 것에 대한 합의된 범위를 설계에서 다루어야 합니다.

파일럿이 이전 POC 를 기반으로 하는 경우 원래 디자인을 다시 상기하도록 주의를 기울여야 합니다. 위에서 언급한 바와 같이, 편의상 POC 는 솔루션의 견고성을 제한했을 가능성이 높으며 후속 파일럿은 개선이 필요합니다.

11.6. Design Authority

디자인 기관의 주요 활동은 다음을 정의하고 관리하는 것입니다.

- The design process
- The design documentation to be used
- The design review process

설계 책임자는 다음과 같은 일반적인 기술의 사용 방법을 결정해야 합니다.

- Email – email clients or SMTP?
- Workflow systems – Blue Prism 은 기존 워크플로 시스템을 활용하여 수동 사용자와 어떻게 상호 작용할 건인가?
- Third party code – 이것을 어떻게 블루 프리즘에 포장해서 노출시켜야 할 것인가?
- Web services

설계 책임자는 설계 및 개발 규칙(예 : 프로세스, 객체, 작업 대기열, 환경 변수, 데이터 항목 등에 대한 명명 규칙)을 정의해야 하며, 로깅 정책 및 데이터 스토리지 정책을 규정하고 모범 사례 개발의 관리자가 되어야 합니다.

Appendix

12. Design Review Checklist

12.1. Solution

- 'as is' 수동 프로세스를 정의하고 자동화 솔루션을 설계하기에 충분하게 세부 사항을 문서화했습니까?
- 고객의 요구 사항이 문서화되고 동의를 받았습니까?
- 이 디자인이 요구 사항을 충족합니까?
- 설계를 통해 고객이 'to be' 자동화 솔루션을 이해하고 승인할 수 있습니까?

12.2. Process

- '케이스 작업' 단계 외부에는 어떤 로직이 있습니까?
  - '준비(Preparation)' 단계가 필요합니까?
  - '마무리(Finalisation)' 단계가 필요합니까?
  - '재설정(Reset)' 단계가 필요합니까?
  - '복구(Recover)' 단계가 필요합니까?
- 프로세스가 Blue Prism 작업 대기열을 사용합니까?
  - 그렇지 않다면, 그 이유는 무엇입니까?
- 프로세스가 여러 시스템에서 병렬로 실행되는 경우 어떤 효과가 있습니까?
  - 여러 시스템에서 동시에 실행해서는 안되는 단계가 있습니까?
  - 실행중인 머신 수에 관계없이 한 번만 실행해야 하는 단계가 있습니까?
- 프로세스에서 암호는 어디에서 있습니까?
  - 자격 증명 저장소(credential store)에 보관됩니까?
  - 다이어그램에 하드 코딩된 것이 있습니까?
- 암호는 언제 만료됩니까?
  - 비밀번호 변경은 어떻게 관리됩니까?
- 프로세스가 어떻게 중지됩니까?
  - 대기열이 비어 있을 때?
  - 특정 시간에?
- 예외가 발생한 후 응용 프로그램이 다음 사례를 처리하기에 이상적인 상태가 아닐 수 있습니까?
  - 그 상황은 어떻게 정상화될 것입니까?
  - 응용 프로그램을 다시 시작해야 합니까?
- 예외는 어떻게 관리됩니까?

    - 대기열 재시도가 사용됩니까?
    - 예외는 비즈니스에게 어떻게 전송됩니까?
    - 예외 비율은 어떻게 모니터링됩니까?
- 대기열 결과를 다른 Blue Prism 대기열, 워크플로 응용 프로그램, 데이터베이스 또는 파일에 복제해야 합니까?
    - 어떻게 그리고 언제 완료됩니까?
    - 양측이 불균형해질 가능성이 있습니까?
- 알림을 보내는데 프로세스가 필요합니까?
    - 이 작업은 언제 어떻게 수행됩니까?
- 하위 프로세스를 사용합니까?
    - 메모리 누수의 위험이 있습니까?

12.3. Objects

- 이 솔루션에 필요한 객체가 항목별로 분류되었습니까?
    - 이미 존재하는 객체와 생성해야 하는 객체가 분명히 구분됩니까?
    - 개발 작업을 시작하기 전에 설계를 검토할 수 있는 체크 포인트가 있습니까?
- 이 솔루션을 위해 생성된 객체를 재사용할 수 있습니까?
    - 그렇지 않다면 그 이유는 무엇입니까?
- 재사용을 제약할 수 있는 '비즈니스 프로세스 로직'이 객체 계층에 있습니까?
    - 이 로직이 프로세스 레이어에 있어야 합니까?
- 시스템 예외 이외에 프로세스에서 특별한 처리가 필요한 예외가 발생하는 개체가 있습니까?
    - 이것은 객체가 사용자에게 명확합니까?
    - 출력 매개 변수가 오히려 더 명시적이지 않을까요?
- 나눌 수 있을 만큼 지나치게 복잡한 페이지가 있습니까?
    - 사용 편의성을 위하여
    - 쉬운 재사용을 위하여
    - 더 효과적인 테스트를 위하여
    - 효율성을 증대하기 위하여
    - 더 나은 보안을 위하여
- 객체와 페이지의 이름이 프로세스 개발자에게 목적에 대한 좋은 직관을 제공합니까?
    - 무의미하거나 모호한 이름이 있습니까?