Solution Design Overview

TRAINING GUIDE

VERSION: 1.00

Blue Prism Limited, 2 Cinnamon Park, Birchwood, WA2 0XP, United Kingdom
Registered in England: Reg. No. 4260035. Tel: +44 870 879 3000. Web: www.blueprism.com

Contents

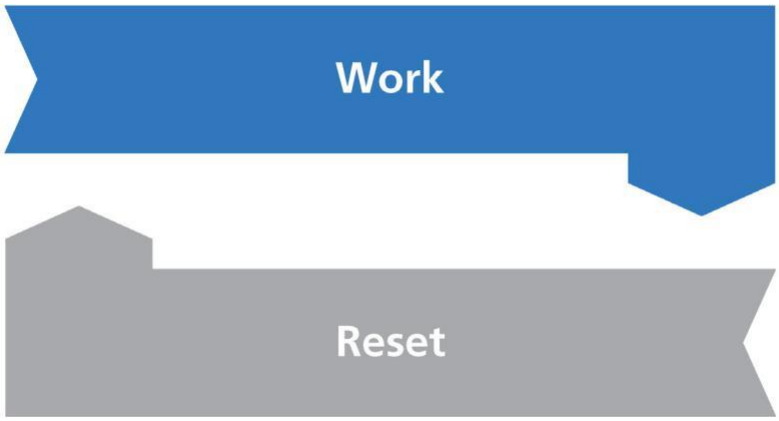
1. Introduction

Blue Prism 은 자동화된 프로세스를 생성하고 안전한 가상 환경에서 실행하기 위한 플랫폼입니다. 자동화된 프로세스는 무인으로 감시하지 않고도 실행될 수 있어야 합니다.

2. Designing for Unattended Automation

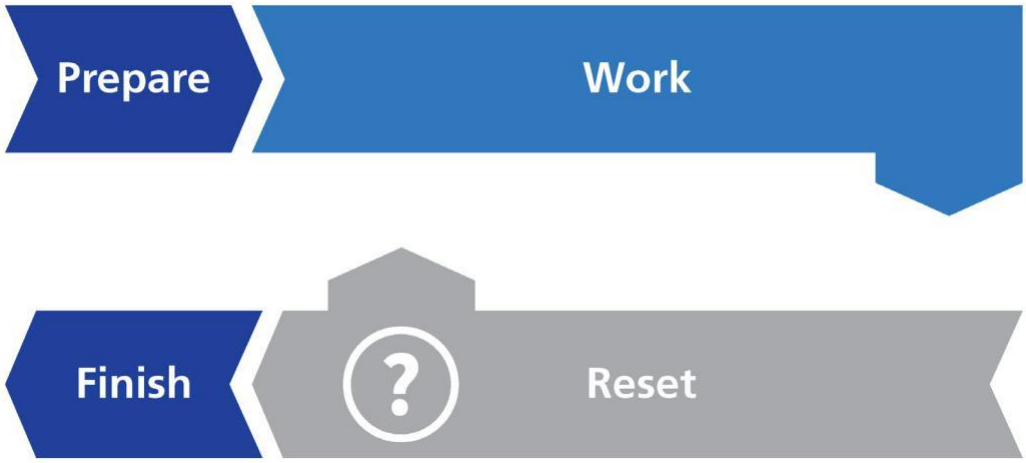
Blue Prism 프로세스는 여러 개의 일련의 수작업 단계를 자동화하는 데 필요합니다. 무인으로 작업하고 예기치 않은 상황에서 복구하고 여러 시스템에서 동시에 작업할 수 있을 만큼 견고하게 만들어야 합니다. Blue Prism 솔루션을 설계하는 방법을 배울 때 이렇게 구현하는 것을 종종 놓치곤 합니다.

프로세스의 목적은 케이스를 하나씩 처리하는 것이며, 한 케이스를 완료한 후에는 다음 케이스를 준비하기 위해 시작 위치로 돌아가야 하기 때문에 이것은 너무 단순합니다. 이 프로세스는 사용 중인 응용 프로그램과 데이터에 대한 제어를 유지해야 하므로 일부 재설정 단계를 수행해야 할 수 있습니다. 이러한 재설정 단계에는 애플리케이션의 기본 메뉴 화면으로 다시 이동하고 데이터 항목 값을 재설정하는 작업이 포함될 수 있습니다.

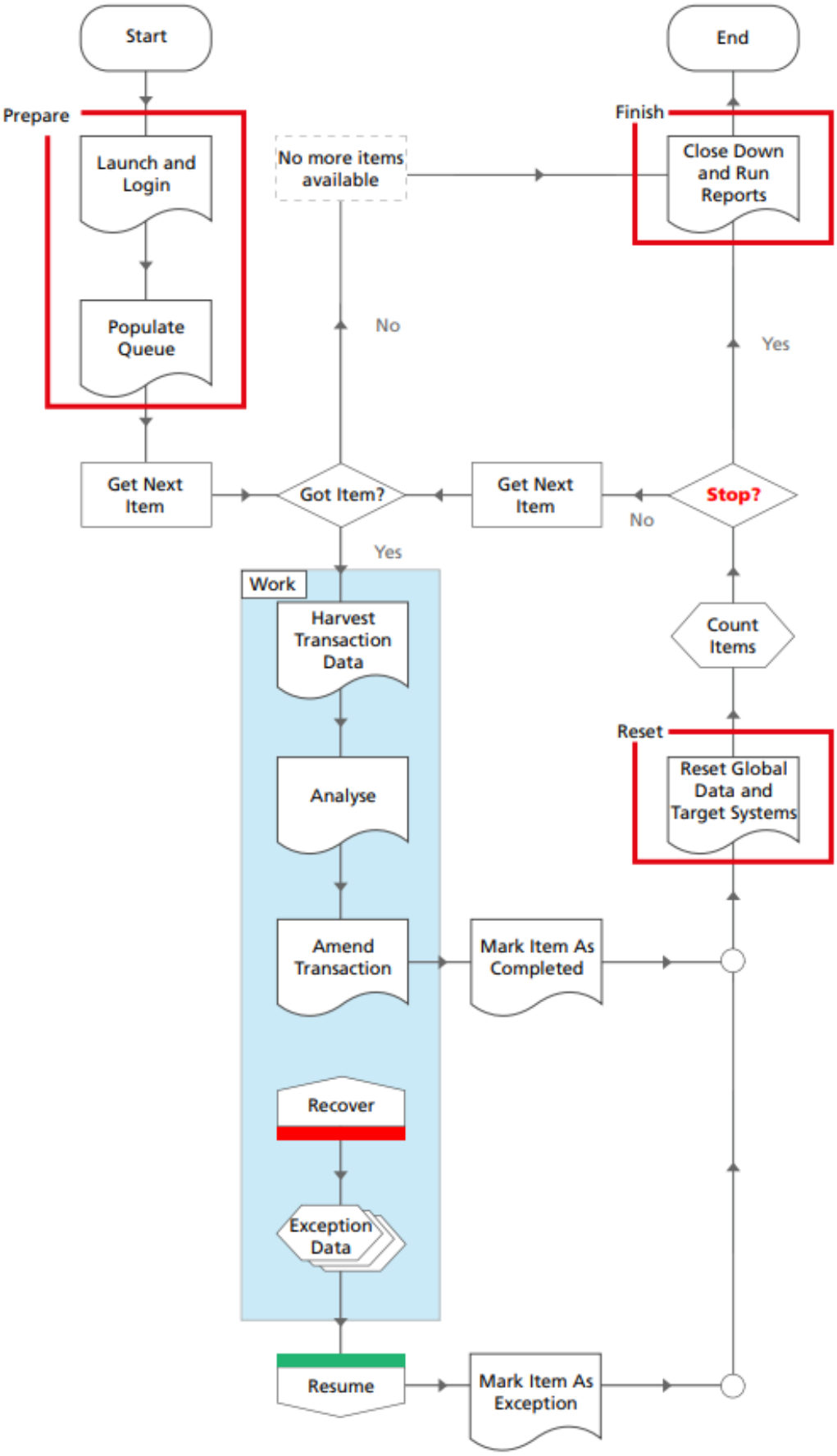


어떤 시점에서 프로세스는 이 루프에서 벗어나야 합니다. 종종 이는 작업할 케이스가 더 이상 남아 있지 않을 때이지만 다른 이유로 결정을 내릴 수도 있는 경우입니다. 예를 들면 작업 일이 끝날 때에 도달했을 경우입니다. 또한 작업 루프 전후에 추가 단계가 있을 수도 있습니다.

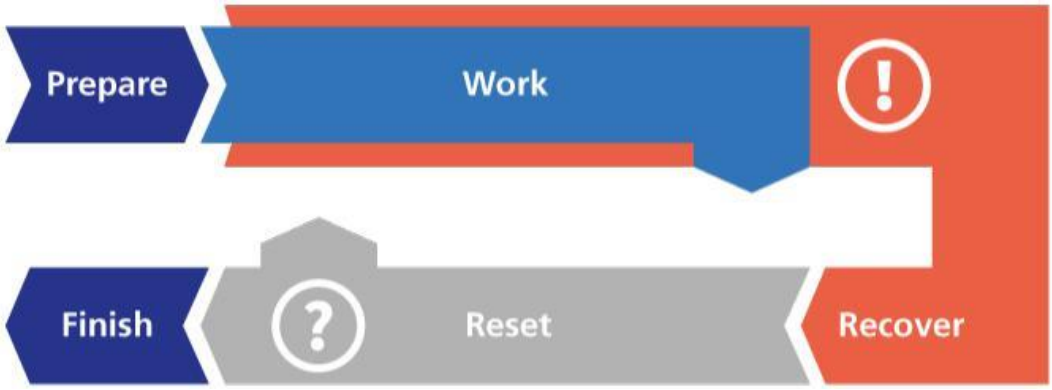
준비 단계의 예는 애플리케이션 로그인 및 입력 데이터 수집과 같은 것입니다. 마무리 단계는 로그아웃 및 애플리케이션 종료, 보고서 작성 또는 이메일 알림 전송과 같은 것일 수 있습니다.



이를 설명하기 위해 Blue Prism 기본 템플릿을 사용하여 구성된 다음 프로세스를 생각해 보겠습니다. 케이스 처리(작업) 외부의 세 개의 영역이 명확하게 표시됩니다.


Start
End
Prepare
Finish
Launch and Login
No more items available
Close Down and Run Reports
Populate Queue
No
Yes
Get Next Item
Got Item?
Get Next Item
Stop?
No
Yes
Work
Harvest Transaction Data
Count Items
Reset
Analyse
Reset Global Data and Target Systems
Amend Transaction
Mark Item As Completed
Recover
Exception Data
Resume
Mark Item As Exception

프로세스를 설계할 때 또 다른 일반적으로 간과하는 것은 문제가 발생하지 않고 항상 '행복한 경로'를 유지한다고 가정하는 것입니다. 다시 말해서 이것은 비현실적이며 복구 단계는 아래의 복구 경로에 표시된 것처럼 '불완전한 경로' 논리가 처리를 위해 꼭 포함되어야 합니다.



여기서 예외 처리는 예기치 않은 애플리케이션 동작에서 프로세스를 복구하는 데 사용되므로 케이스를 수동 조사하기 위해 따로 설정하거나 필요한 경우 프로세스에 의해 재작업을 할 수 있습니다. 복구 논리에는 프로세스가 계속 작동하도록 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 다음과 같은 추상 프로세스를 고려하십시오.

3.1 Full Automation

이상적인 시나리오는 모든 수동 단계를 자동화하고 전체 수동 프로세스를 교체할 수 있는 경우입니다. 단순성을 위해 예외 사례를 수동으로 해결하는 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. 이상적으로는 나머지 수동 단계의 노력은 자동화된 단계로 인한 절감 효과로 상쇄됩니다.

종종 이러한 종류의 부분 자동화에서는 자동화를 활성화하기 위해 나머지 수동 단계를 조정해야 합니다. 예를 들어, 단계 M1 에서 사용자는 단계 A2 가 소비할 구조화된 데이터를 준비해야 할 수 있습니다. 유사하게, 단계 M9 에서 사용자는 A8 로부터 핸드 오프의 수신자가 될 필요가 있을 수 있습니다.

3.3 Fragmented Partial Automation

연속 자동화를 달성할 수 없는 경우 자동화를 여러 단계 시퀀스에 적용할 수 있습니다.

여기에서 전체적인 이점을 평가할 때 수동 단계와 자동 단계 간의 연결을 고려해야 합니다. 핸드 오프에 새로운 수동 작업이 필요한 경우 자동화 아이디어가 무효화될 수 있지만 동기화된 상호 작용이 쉽게 달성될 수 있다면 단편화된 자동화가 작동할 수 있습니다.

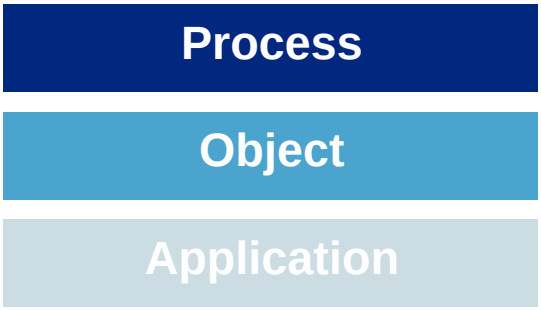
3.4 Restructured Partial Automation

단편화를 방지하고 지속적인 자동화 시퀀스를 용이하게 하기 위해, 수동 단계의 일부를 다시 엔지니어링하거나 재정렬할 수도 있습니다.

여기서 1, 4, 5 단계는 2, 3, 6, 7, 8 을 하나의 직접적인 프로세스로 자동화할 수 있도록 수동 순서로 재배열되었습니다.

4. Solution Layers

Blue Prism 자동화 솔루션은 프로세스 계층에서 시작하여 객체 계층과 애플리케이션 모델, 그리고 애플리케이션 계층까지 단계적 계층으로 인식될 수 있습니다.

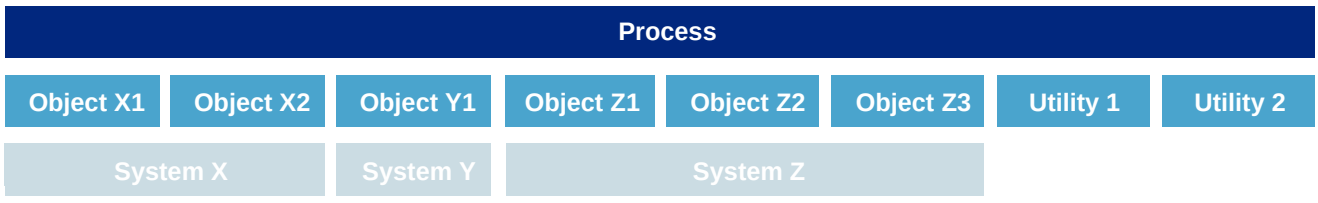


프로세스 계층과 객체 계층의 역할은 처음에는 혼란스러울 수 있지만 본질적으로 객체는 응용 프로그램을 조작하는 메커니즘을 프로세스에 제공하는 도구로 간주되어야 합니다.
일반적으로 비즈니스 로직, 규칙 및 의사 결정은 프로세스 안에 위치해야 합니다.

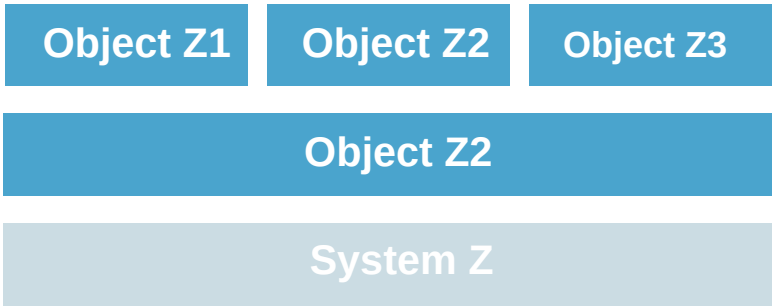
4.1 Business Objects

비즈니스 오브젝트는 프로세스가 애플리케이션을 제어하는 데 사용하는 도구입니다. 이상적으로 객체는 프로세스가 복잡한 시퀀스로 오케스트레이션할 수 있는 일련의 간단한 기능을 제공해야 합니다. 비즈니스 규칙 및 의사 결정에 대한 책임의 대상을 제거함으로써 목표는 다른 프로세스에서 객체를 재사용하고 객체 '라이브러리'를 구축할 수 있도록 하는 것입니다. 그리고 객체 라이브러리의 다양성이 증가함에 따라 자동화 솔루션을 제공하려는 노력도 줄어들어야 합니다.

자동화된 솔루션의 객체 계층은 둘 이상의 객체로 구성될 가능성이 높으며 종종 동일한 애플리케이션의 다른 측면을 자동화하는데 여러 객체가 사용됩니다. 응용 프로그램과 관련이 없는 일반 기능을 제공하는 데 사용되는 유틸리티 객체도 포함될 수 있습니다.



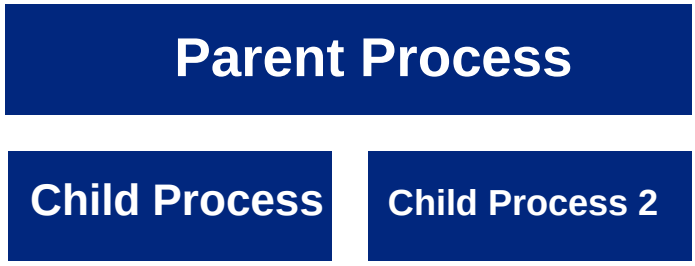
객체는 다른 객체를 사용할 수도 있으며 Surface Automation 과 같은 일부 상황에서는 다른 객체가 사용하는 대상 시스템과 상호 작용할 '기본' 객체를 만드는 것이 바람직할 수 있습니다.



모든 객체가 대상 시스템과 상호 작용할 필요는 없습니다. 예를 들어 여러 프로세스에서 사용할 수 있는 규칙 엔진이 필요할 수 있습니다. 이것은 시작 매개 변수를 통해 데이터를 받아들이고 출력 매개 변수를 통해 결정을 다시 전달하기 전에 데이터를 조사하는 동작이 있는 객체에서 개발될 수 있습니다.

4.2 Sub-processes and Wrapper Objects

프로세스는 다른 프로세스를 자식으로 호출하여 프로세스 계층을 만들 수도 있습니다.

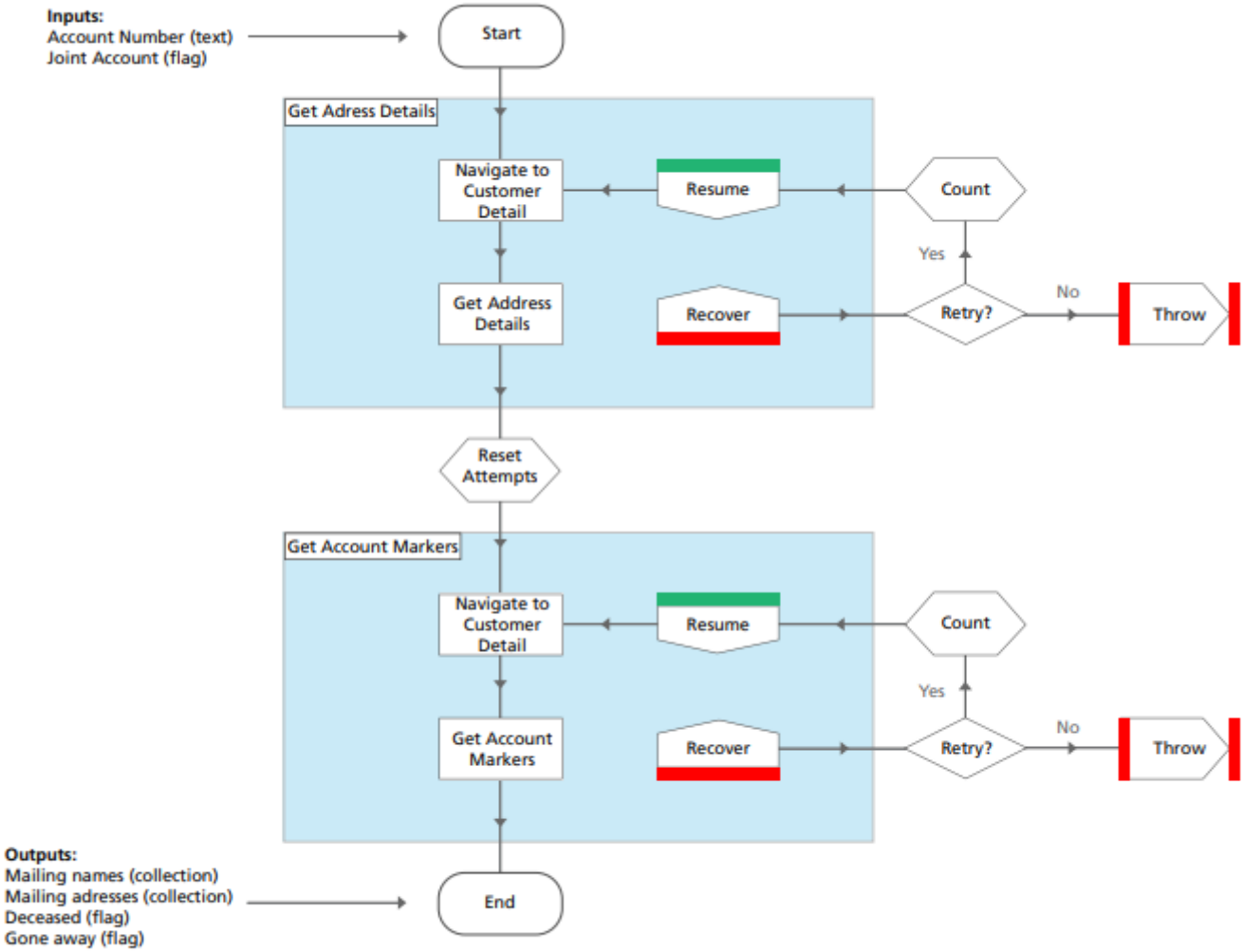


그러나 Blue Prism 이 메모리를 관리하는 방식은 자식 프로세스 사용을 고려할 때 신중하게 고려해야 합니다. 객체와 달리 자식 프로세스는 부모가 메모리에 보존하지 않습니다. 자식 프로세스가 종료되면 부모 프로세스는 .Net Garbage Collector 가 회수할 메모리를 해제합니다. 이 때문에 부모가 자식 프로세스를 반복적으로 사용하는 경우 가비지 콜렉터가 정리할 수 있는 것보다 더 빨리 자식 프로세스(및 해당 객체)에서 메모리를 사용하면 원하지 않는 메모리 누수가 발생할 수 있습니다.

따라서 반복적으로 호출되는, 예를 들면 작업 루프안에서, 하위 프로세스를 사용하지 않는 것이 좋습니다. 대안으로 객체를 사용하여 다른 객체의 동작을 래핑합니다.

예를 들어, 편지를 보내기 전에 고객의 우편 주소 세부 정보를 수집할 때 공통된 단계가 있는 경우가 그것입니다. 고객 서신을 주문하는 각 프로세스에서 이를 복제하는 대신 모든 프로세스에서 호출할 수 있는 독립 실행형 객체로 모든 작업을 함께 래핑할 수 있습니다.

아래 작업은 대상 시스템과 인터페이스하지 않는 객체에서 수행한 것입니다. 그러나 세 가지 다른 객체에서 수행하는 네 가지 작업을 래핑합니다.



5. Solution Layers

5.1 Recoverability

간단히 말해서, 복구 가능성은 문제를 처리하고 정상 상태로 돌아가는 솔루션의 능력입니다. 행복한 길이 유일한 가능성이라고 가정하는 것은 순진한 것이니, 예기치 않은 상황에서 복구를 시도할 수 있는 준비가 항상 이루어져야 합니다.

System Recovery

자동화된 솔루션이 항상 애플리케이션에 대한 제어를 유지하는 것이 중요합니다. 모든 것이 순조롭게 진행될 때 이것은 비교적 간단하며 확인해야 할 유일한 것은 다음 케이스를 처리하기 전에 애플리케이션이 시작 위치로 돌아간다는 것입니다.

응용 프로그램이 예상대로 작동하지 않고 제어 기능이 상실되면 어려움이 따릅니다.

일반적으로 이것은 객체의 '시간 초과' 또는 '요소를 찾지 못함' 오류로 나타납니다.

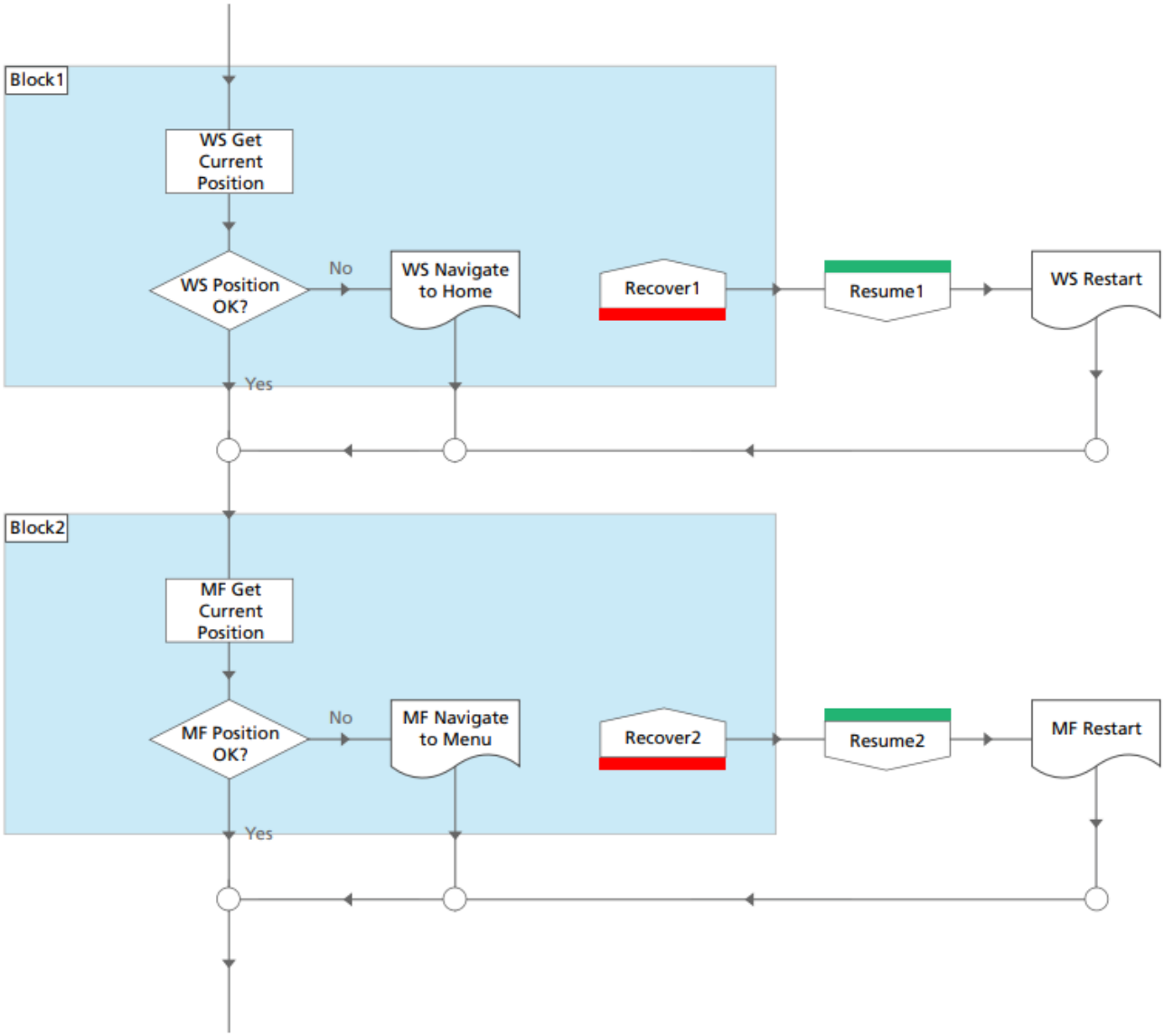
결과 예외는 예외 처리로 관리할 수 있지만 프로세스를 계속하려면 응용 프로그램을 다시 제어해야 합니다.

일부 애플리케이션에서는 이 작업이 쉬울 수 있습니다. 예를 들어 웹 애플리케이션에서는 홈 URL 로 직접 이동하는 경우일 수도 있고 메인 프레임에서는 모든 위치에서 되돌릴 수 있는 범용 키 입력이 있을 수 있습니다. 하지만 뒤로 이동하는 것이 그렇게 간단하지 않은 경우도 있으며 애플리케이션을 다시 시작해야 할 수도 있습니다. 다시 말하지만 이것은 고통스럽지 않을 수 있으며 응용 프로그램을 종료하고 다시 시작할 수 있습니다. 그러나 일부 응용 프로그램은 대략적인 처리를 원하지 않으며 규정된 로그 아웃 절차를 엄격하게 준수해야 합니다.

각 애플리케이션은 솔루션에 복구 가능성 섹션을 포함할 것으로 예상하여 신중하게 산정해야

합니다. 시스템 복구가 솔루션 설계 문서 또는 프로세스 설계 지침에 명시적으로 캡처되지는 않았지만 개발자는 강력한 솔루션을 구성할 때 개념을 이해해야 합니다.

다음은 다음 사례를 처리하기 전에 두 개의 애플리케이션(웹 시스템 및 메인 프레임)을 준비하는 데 사용되는 복구 논리의 예입니다. 응용 프로그램의 현재 위치가 결정되고 예상과 다르거나 시도가 실패하면 논리가 복구를 시도합니다. '탐색' 페이지는 예외를 발생시키고 재시작을 트리거하기 전에 여러 번 시도합니다. '다시 시작' 페이지도 여러 번 시도하지만 예외가 발생하는 경우 종료를 유발하기 위해 의도적으로 처리되지 않은 상태로 유지됩니다. 바람직하지 않지만 여기에서 종료는 제어가 손실되었으며 주의 없이 처리를 계속할 수 없음을 보여주기 위해 필요합니다.


Block1
WS Get Current Position
WS Position OK?
No
WS Navigate to Home
Recover1
Resume1
WS Restart
Yes
Block2
MF Get Current Position
MF Position OK?
No
MF Navigate to Menu
Recover2
Resume2
MF Restart
Yes

Case Recovery

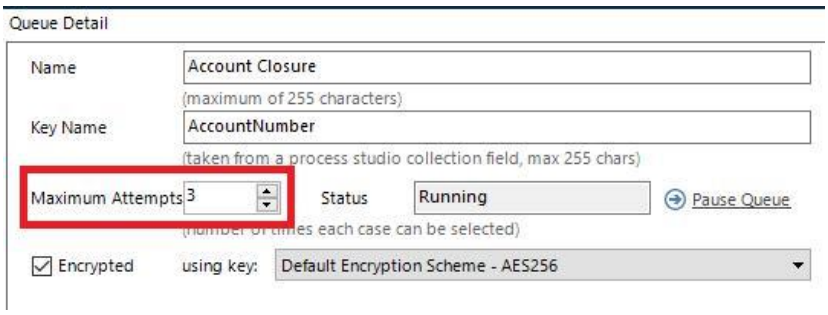
케이스 복구는 프로세스가 다시 시작된 후 케이스를 해결하는 기능입니다. 예를 들어 외부 문제로 인해 프로세스가 케이스를 진행하는 동안 Blue Prism 이 완전히 실패하는 경우 해당 케이스는 어떻게 됩니까? 예외로 수동 팀에 넘겨 질 것이라고 간단히 말하면 충분합니까? 또는 프로세스에 이전 위치에서 사례를 계속할 수 있는 기능이 있어야 하며, 그렇다면 어떻게 달성할 수 있습니까?

사례가 진행될 때 대기열 항목을 업데이트하여 사례의 현재 상태를 기록할 수 있습니다. 그런 다음 갑작스런 쇼크가 발생한 경우 Blue Prism 프로세스 또는 수동 팀에 의해 케이스가 다시 픽업되면 이전 수준의 진행 상황이 알려집니다.

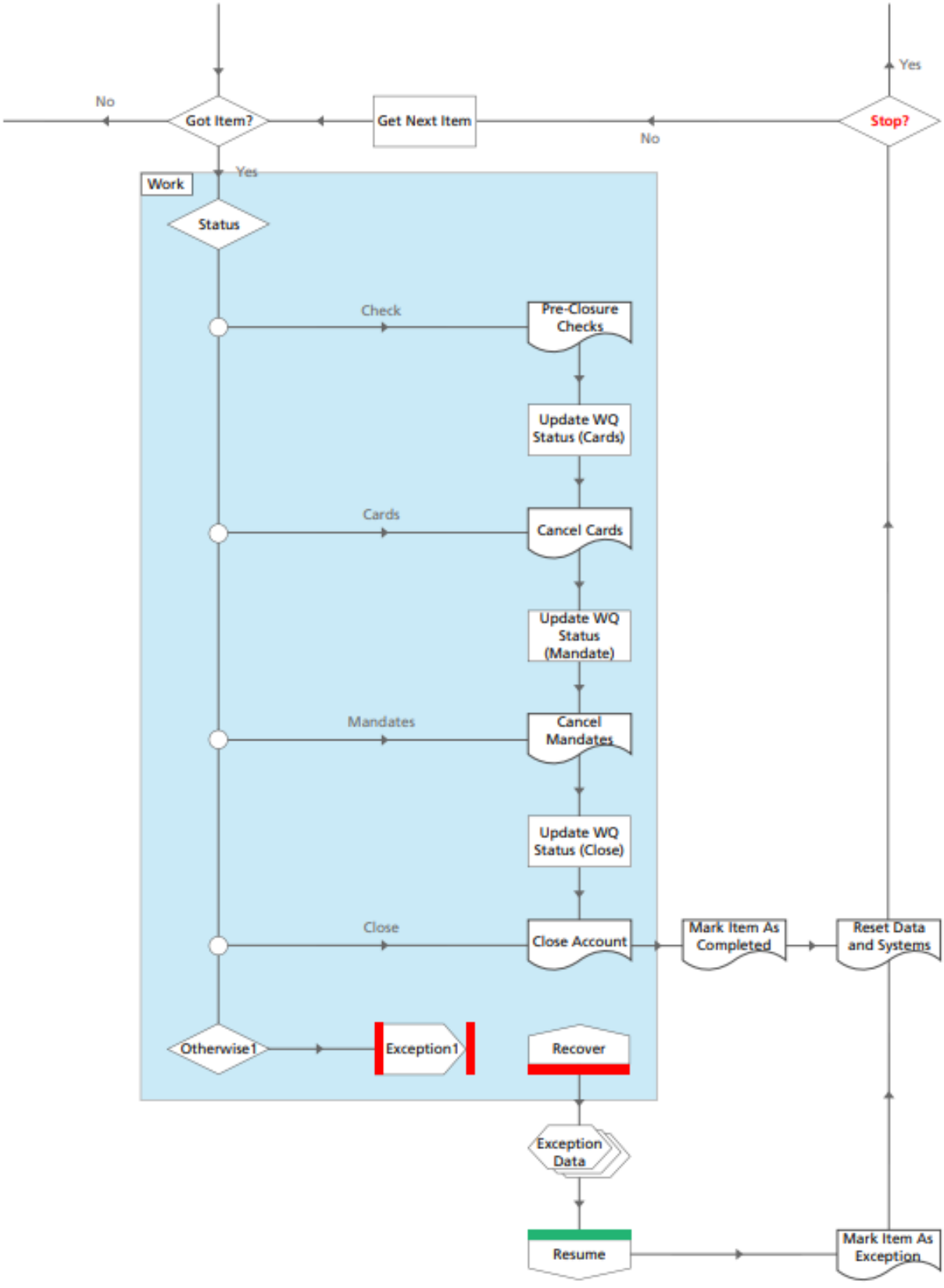
아래 요약에서는 3 단계 이후 문제가 발생할 때까지 각 단계에서 케이스의 상태를 기록합니다.

케이스가 재작업되면 4 단계에서 시작할 수 있습니다.

이를 설명하기 위해 다음 예를 고려하십시오. 계정 폐쇄 작업 대기열은 시스템 관리자에서 케이스 당 최대 세 번의 시도를 허용하도록 구성되었습니다. Blue Prism 은 이 속성을 사용하여 예외로 표시된 대기열 항목을 복제하여 새 보류 항목을 만듭니다.



아래의 계정 폐쇄 프로세스는 대기열 항목 상태 속성을 사용하여 케이스 진행 상황을 추적하도록 설계되었습니다. 프로세스에는 4 단계가 있습니다. 확인, 카드, 위임장 및 마감을 하고 각 단계가 성공적으로 수행되면 상태가 업데이트됩니다. 이렇게 하면 프로세스에 표시된 모든 재시도 사례가 올바른 다음 단계로 이동합니다. 또한 수동 검토를 위해 예외가 전송되면 케이스가 실패한 수동 팀에 상태 값이 표시됩니다.



Database Recovery

프로덕션 데이터베이스를 복구하는 방법은 솔루션 설계의 일부로 고려해야 할 수 있습니다. 프로덕션 데이터베이스가 실시간으로 복제되거나 미러링되지 않는 경우 재해에서 복구하기 위해 백업을 복원해야 할 수 있습니다. 이런 일이 발생하면 백업을 수행한 후 작업 한 사례가 대기열에 작업되지 않은 것으로 나타날 수 있으며 이러한 사례를 재 작업하려는 솔루션의 효과를 고려해야 합니다.

1000 개의 케이스가 작업 대기열에 로드되고 처리 중간에 심각한 데이터베이스 오류가 발생한 예를 고려하십시오. 백업 서비스는 4 시간마다 증분 백업을 제공하지만 최신 백업이 복원되면 모든 케이스가 보류 중으로 표시됩니다. 단순히 프로세스를 다시 시작하면 이미 작업한 500 개의 케이스가 재작업됩니다.

오래된 프로덕션 데이터베이스를 복원하는 효과를 완화하는 것은 설계 단계에서 가장 잘 달성됩니다. 프로세스에 따라 대상 응용 프로그램에 자연적인 방어가 있을 수 있습니다. 예를 들어, 고객 계약을 취소하는 프로세스에서 대상 애플리케이션이 이미 취소된 계약을 다시 취소하도록 허용하지 않은 경우 해당 케이스를 완료로 표시하도록 프로세스를 설계할 수 있습니다.

그러나 자금을 이체하는 금융 프로세스의 예에서 케이스를 재처리하면 재앙적으로 중복 지불을 할 수 있습니다. 그러나 프로세스가 예를 들어 주요 단계에서 계정에 인식 가능한 메모를 적용하여 '발자국'을 찾고 생성하도록 설계된 경우 발자국이 존재하는지 확인하고 중요한 단계를 반복하지 않도록 할 수 있습니다.

5.2 Scalability

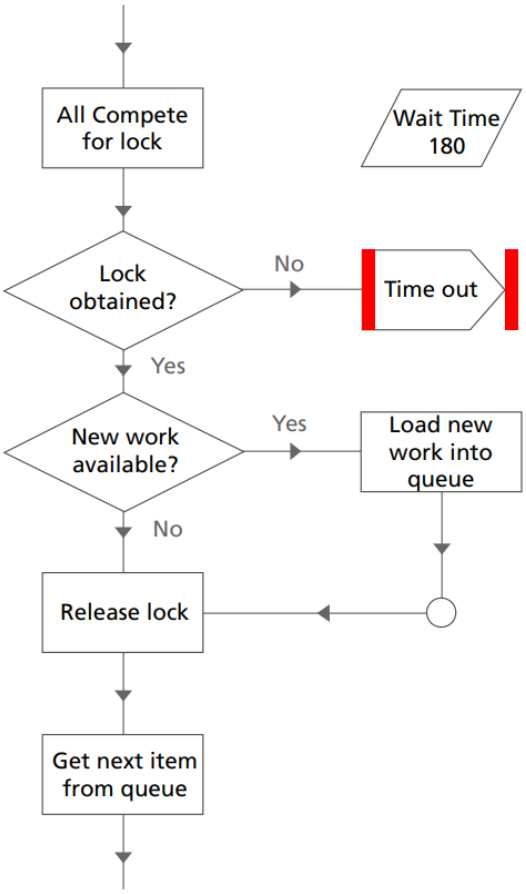
자동화된 프로세스를 설계할 때 일반적으로 간과하는 문제는 프로세스가 여러 시스템에서 동시에 실행되어야 한다는 사실을 잊는 것입니다. Blue Prism 작업 대기열은 여러 프로세스 인스턴스가 동일한 대기열 항목에 액세스하는 것을 방지하지만, 프로세스가 둘 이상의 시스템에서 실행되는 경우 케이스 작업 루프 외부의 단계에 특별한 주의가 필요할 수 있습니다.

예를 들어, 일일 입력 파일을 사용하고 작업 대기열을 로드하여 프로세스를 시작하는 일반적인 시나리오를 생각해 보십시오. 세 개의 프로세스 인스턴스가 실행 중이면 어떻게 됩니까? 그들은 모두 파일을 읽고 큐를 삼중으로 로드하려고 할까요? 또는 하루가 끝날 때 보고서가 작성된다고 가정하십시오. 중복 보고서가 생성되는 것을 어떻게 방지합니까?

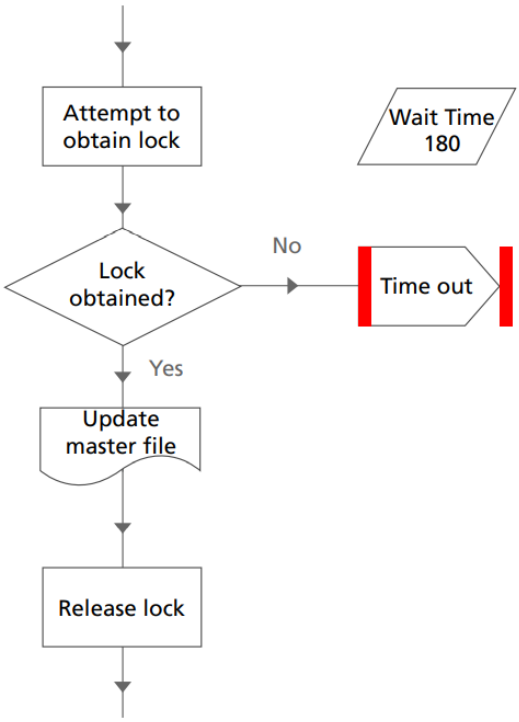
Environment Locks

환경 잠금은 프로세스(및 객체) 간에 '권한'을 공유할 수 있도록 하는 Blue Prism 기능입니다.

하나의 인스턴스만 특정 단계를 수행하도록 권한을 얻기 위해 경쟁하도록 프로세스 인스턴스를 만들 수 있습니다. 예를 들어 프로세스의 첫 번째 단계가 데이터 소스를 읽고 작업 대기열을 로드하는 것이라고 가정해 보십시오. 프로세스의 세 가지 인스턴스가 동시에 시작되고 모두 환경 잠금을 소유하기 위해 경쟁합니다. 승자는 하나만 있을 수 있으며 해당 인스턴스는 대기열을 로드하고 패자는 대기합니다. 잠금이 해제되면 패자는 잠금을 위해 다시 경쟁할 수 있습니다.



또 다른 요구 사항은 특정 단계를 동시에 수행할 수 있는 인스턴스 수를 제어하는 것입니다. 예를 들어 두 명 이상의 사용자가 열 수 없는 파일을 업데이트하는 프로세스를 상상해 보십시오. 환경 잠금은 파일에 대한 '한 번에 하나씩' 접근을 보장하고 프로세스의 인스턴스가 파일을 업데이트하기 위해 차례를 기다리도록 하는 데 사용됩니다.



5.3 Reusability

재사용성은 Blue Prism RPA 전달 방법론의 핵심 원칙입니다. 객체, 래퍼 객체 및 하위 프로세스를 신중하게 설계함으로써 재사용 가능한 논리 라이브러리를 구축하고 전달 노력을 줄이고 유지 관리 오버 헤드를 최소화할 수 있습니다.

객체에는 '비즈니스 프로세스 논리'가 없어야 하며 객체 페이지는 작고 간단하고 기계적인 단계를 실행해야 합니다. 객체의 기능은 응용 프로그램을 조작할 수 있는 도구를 프로세스에 제공하는 것이며, 객체 논리가 더 일반적일수록 다른 프로세스에서 다시 사용할 수 있습니다.

예를 들어 비즈니스 프로세스의 첫 번째 단계가 'MediSys 를 열고 환자 세부 정보 얻기'인 경우 첫 번째 직감은 이를 수행하는 객체 페이지를 만드는 것입니다. 그러나 'MediSys 를 열고 임상의 약속 받기'로 다른 프로세스가 시작된 경우 객체를 변경(다시 테스트)하지 않고는 기존 MediSys 로직을 재사용할 수 없습니다.

더 나은 전술은 각각 기본 단계를 수행한 일련의 페이지를 디자인하는 것입니다. '시작', '로그인', '환자 찾기'및 '환자 세부 정보 읽기'. 이러한 방식으로 두 번째 프로세스는 첫 번째 프로세스에 대해 생성된 로직 중 일부(예: '시작' 및 '로그인')를 활용할 수 있습니다. 따라서 객체 기능의 세분성을 높이고 애플리케이션의 한 화면에서 '원자적' 작업(예: 읽기, 쓰기 또는 탐색)을 수행하는 페이지를 만들면 객체를 더 많이 재사용할 수 있습니다.

일반적으로 큰 객체도 피해야 합니다. 페이지가 많은 객체는 작동하지만 비효율성을 가져올 수 있습니다. 네트워크를 통해 큰 객체를 전송하려면 더 많은 네트워크 대역폭이 필요합니다. 사용자는 얼마나 잠깐 사용하든 상관없이 전체 객체를 PC 메모리에 저장합니다. 한 번에 한 명의 개발자만 객체에 대해 작업할 수 있습니다. 단일 객체에 대한 종속성이 증가함에 따라 여러 프로세스에 영향을 미치는 객체의 오류 영향이 증가합니다.

이러한 문제를 방지하려면 응용 프로그램 통합 논리를 하나의 큰 객체에 집중하는 대신 일련의 작은 객체로 분리하는 것이 좋습니다. 이 접근 방식은 여러 가지 이점을 제공합니다. 가벼운 객체는 대역폭, 메모리 및 디스크 공간을 덜 사용합니다. 객체 그룹을 사용하면 팀에서 응용 프로그램 통합을 쉽게 개발할 수 있습니다. 그리고 손상된 객체의 잠재적인 영향이 최소화됩니다.

자세한 내용은 다음을 검토하시면 샘플 객체 디자인 지침을 찾을 수 있습니다.

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

6. Case Management

6.1 Reset

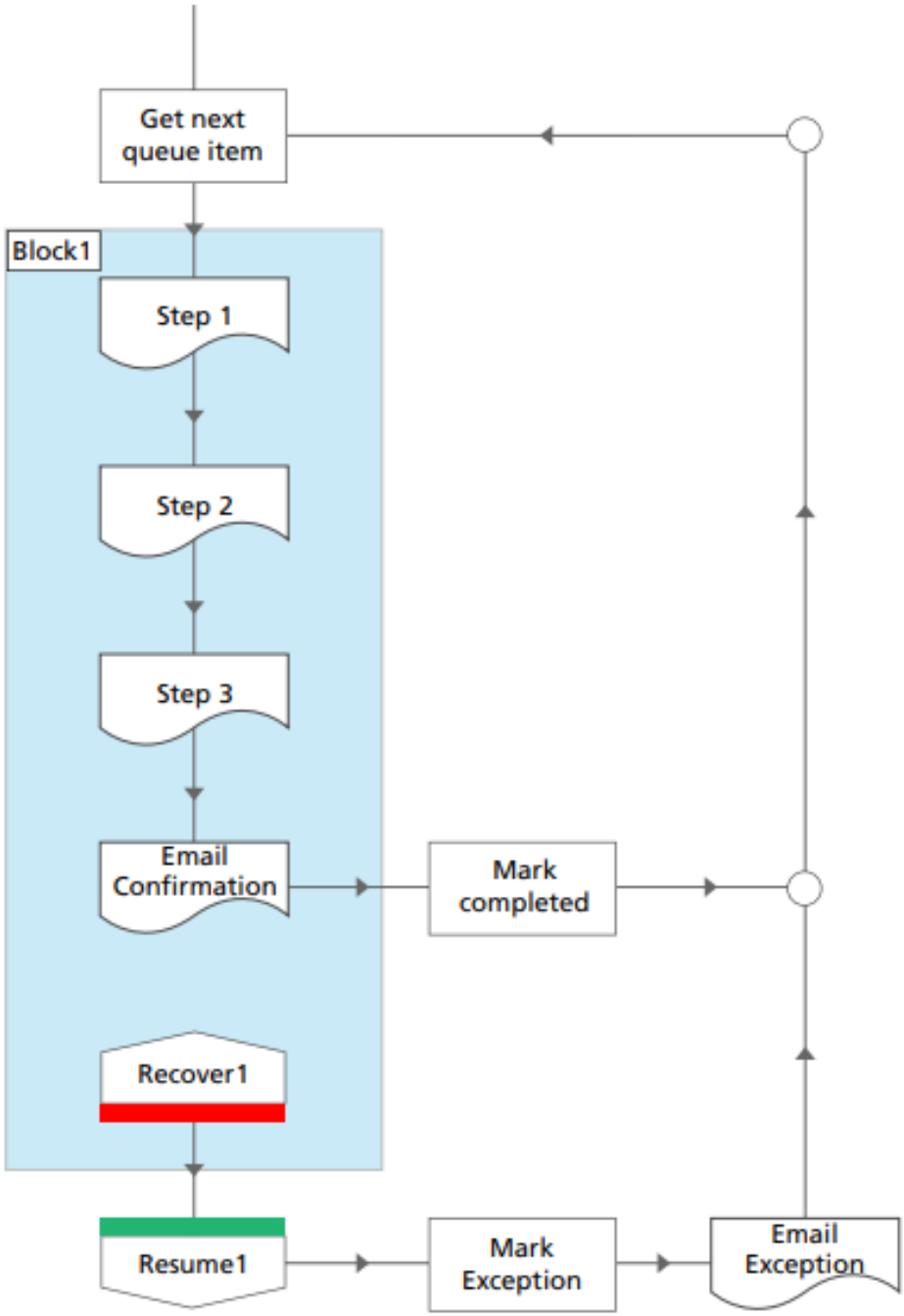
거의 모든 프로세스에는 하나의 케이스가 차례로 작동하는 메인 루프가 포함됩니다. 데이터 항목은 재사용되므로 이전 사례의 값이 다음 사례에서 실수로 사용되지 않도록 주의해야 합니다.

위에서 언급했듯이 프로세스는 살아있는 동안 객체를 메모리에 보관하므로 객체 계층 내의 데이터 값이 유지될 수 있습니다. 이는 데이터가 대소 문자별로 다르지 않은 경우 바람직할 수 있지만 현재 케이스에 영향을 미치는 이전 케이스 데이터를 피하는 것이 중요합니다.

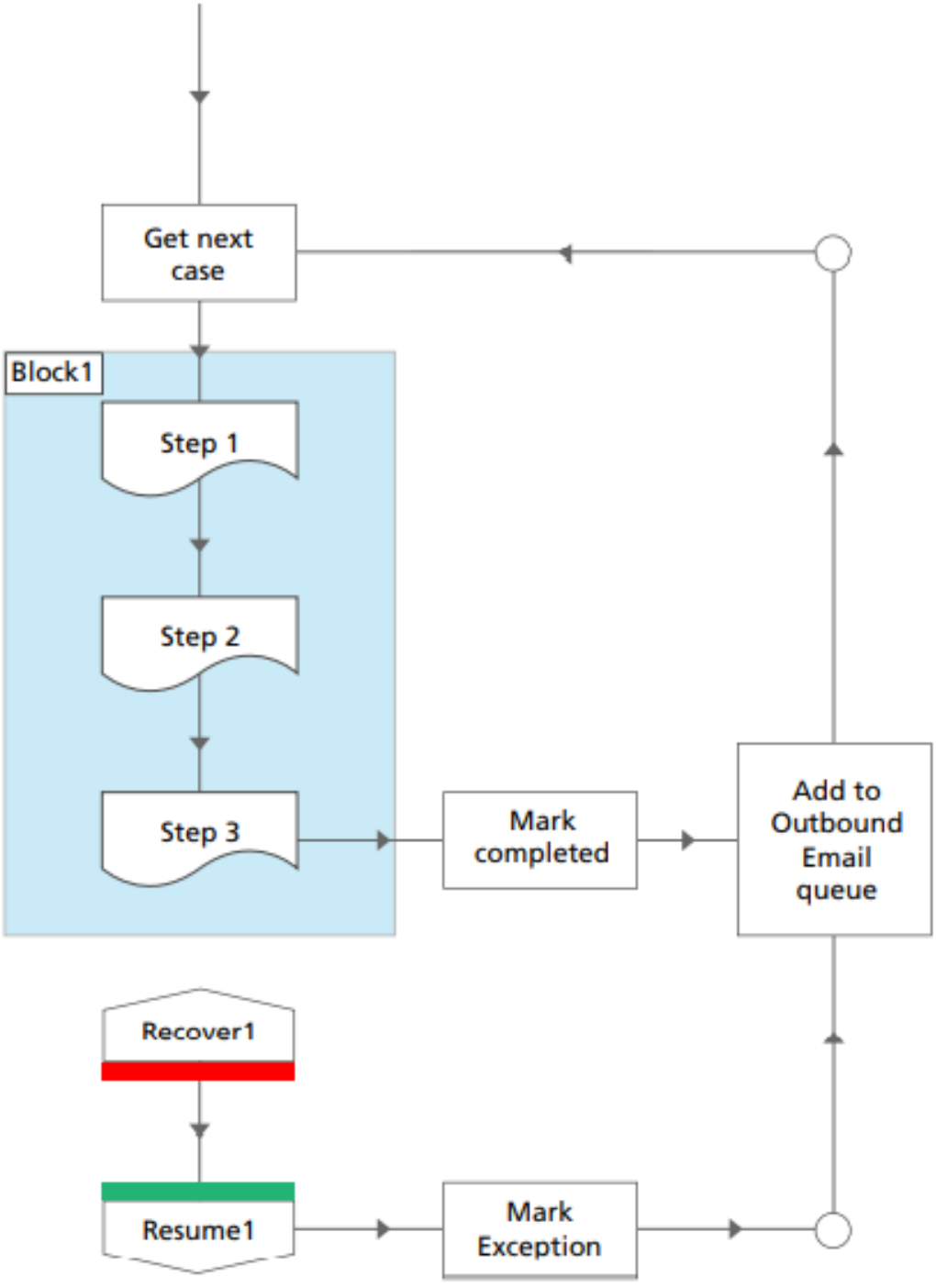
6.2 Resilience

다시 시작한 후 사례를 복구하는 기능과 마찬가지로 솔루션 설계는 복원력을 향상시키고 예외 사례 수를 줄일 수 있습니다. 여러 대기열을 사용하여 사례를 일련의 작업으로 세분화할 수 있습니다. 예를 들어, 이메일이나 웹 서비스를 통해 각 사례가 끝날 때 결과를 전달해야 하는 경우 이 작업을 보조 대기열에서 별도의 사례로 간주하는 것이 좋습니다. 이렇게 하면 필요한 경우 작업을 나중에 완료하거나 다른 컴퓨터에 위임할 수 있습니다.

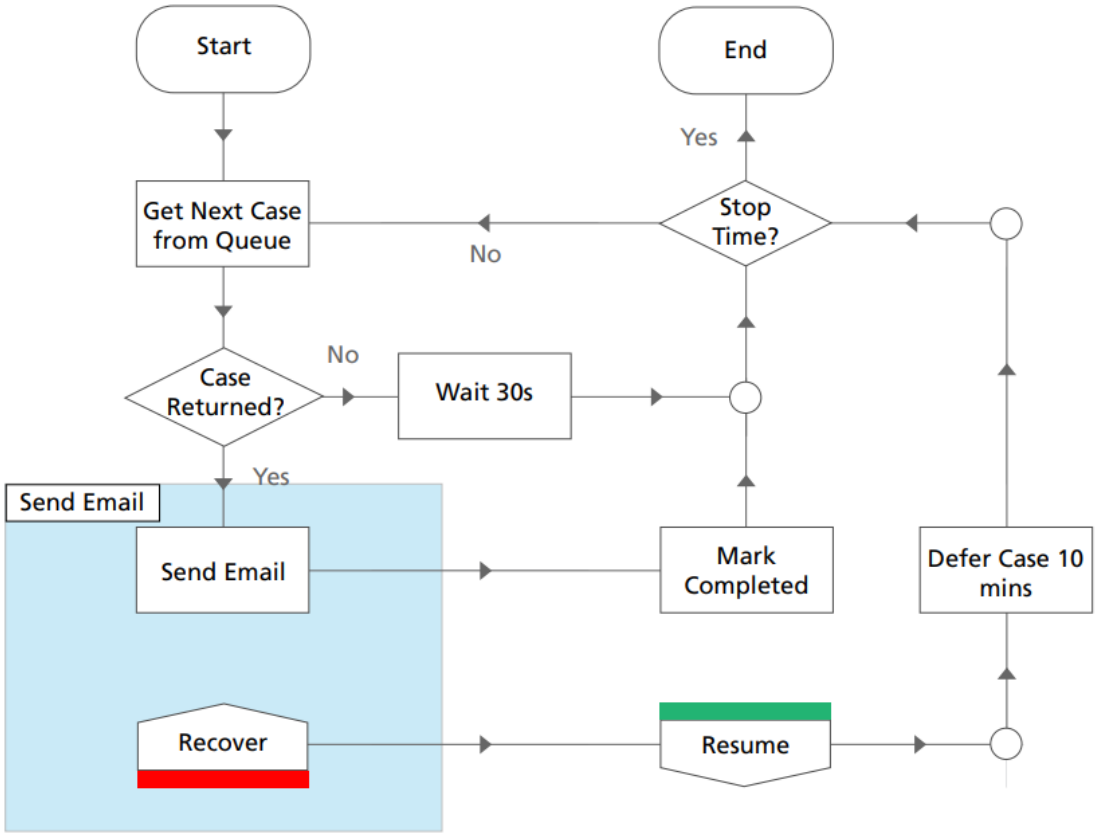
이(의도적으로 단순화되었지만) 예에서 프로세스는 사례를 처리하고 결과를 이메일로 보냅니다. 예를 들어 메일 서버를 사용할 수 없어 확인 이메일을 보낼 때 예외가 발생하면 사례가 예외로 표시되고 수동 검토를 위해 전송됩니다. 케이스가 본질적으로 완료되었다면 이것은 불필요합니다.



그러나 아래의 예에서 프로세스는 이메일을 보내지 않고 대신 다른 프로세스가 보낼 수 있도록 작업 대기열에 메시지를 추가합니다.



별도의 '이메일 보내기' 프로세스는 다른 프로세스와 동시에 작동합니다. 이 프로세스는 항목을 대기열에 추가하는 다른 프로세스를 대신하여 이메일을 보낼 수 있습니다. 이메일 전송에 실패한 경우 사례가 연기되고 나중에 재시도됩니다. 이메일 게이트웨이에 심각한 문제가 있는 경우 문제가 해결되면 이메일이 전송됩니다.



7. Workload Management

7.1 Accountability

거의 모든 자동화 솔루션은 파일 읽기, 받은 편지함 폴링, 웹 서비스 등과 같은 일종의 워크로드를 소비합니다. 그리고 필연적으로 이 데이터는 항상 감사 및 MI 를 위해서 뿐만 아니라 프로세스의 여러 인스턴스 간에 작업을 안전하게 배포하는 수단으로 Blue Prism 작업 대기열에 로드됩니다.

이를 위해 모든 작업을 설명할 수 있고, 사례를 손실하지 않고, 중복시키지도 않으며, 모든 결과를 입증할 수 있는 솔루션을 설계할 수 있어야 합니다. 일단 구현되면 솔루션은 더 넓은 기업 운영의 일부가 될 것이므로 제공된 작업을 안정적으로 소비할 뿐만 아니라 신뢰할 수 있는 결과를 경영에 다시 보고하도록 설계되어야 합니다. 최소한 이 커뮤니케이션은 예외 사례 및 추천을 위한 것이지만 완료된 사례의 세부 정보도 포함해야 할 수도 있습니다.

필요한 수준의 책임을 달성하려면 각 사례를 작업하는 동안 수집된 입력 데이터 정보를 업데이트해야 할 수 있습니다. 이것은 큐 항목 상태 필드를 설정하거나 태그를 적용하는 것처럼 간단할 수 있지만 큐 항목 데이터 콜렉션에 데이터를 쓰는 것도 포함할 수 있습니다.

7.2 Balance

워크플로 도구 또는 데이터베이스와 같은 다른 응용 프로그램에서 작업을 가져오기 위해 자동화된 솔루션이 필요한 경우는 드물지 않습니다. 또한 출처로 돌아가 노동 결과를 업데이트해야 할 수도 있습니다. 디자이너는 데이터 소스와 자동화된 솔루션의 균형을 유지하고 케이스가 작동하지 않고 중복이 발생하지 않으며 결과가 반드시 보고되도록 확실히 주의를 기울여야 합니다.

일부 디자인은 여러 Blue Prism 작업 대기열을 사용해야 할 수 있으므로, 균형이 유지되도록 다시 고려해야 합니다.

7.3 Overload

워크로드 급증이 발생할 수 있고/또는 SLA 가 엄격하다면, 임박한 문제를 감지하고 알리는 메커니즘을 설계하는 것이 정당할 수 있습니다. 예를 들어, 보류 중인 항목 수를 평균 케이스 시간과 곱하면 완료 시간 예측을 SLA 와 비교할 수 있습니다.

마찬가지로, 비정상적으로 가볍거나 존재하지 않는 워크로드도 자동 경고를 발생하도록 하거나 대체 조치를 트리거할 수 있습니다.

8. Data Management

8.1 Preservation

기능 요구 사항은 솔루션 데이터를 보존할 기간을 나타내야 하지만, 일종의 제어가 구현되지 않는 한 일부 유형의 데이터가 무기한으로 유지된다는 점을 인식하는 것이 중요합니다. 세션 로그는 System Manager 를 사용하여 보관할 수 있으며 보관 정책을 마련하여 로그 데이터의 증가를 제한해야 합니다.

그러나 작업 대기열 데이터는 자동화된 메커니즘이 생성되지 않는 한 데이터베이스에 남아 있습니다. 이는 단일 'Manage All Queues' 프로세스의 형태이거나 'Internal –Work Queues' 객체의 기능을 사용하여 각 프로세스가 자체 대기열을 유지하도록 할 수 있습니다. 일반적으로 '유지 관리'란, 워크로드, MI 또는 감사의 일부로서, 더 이상 어떤 용도로도 사용되지 않는 오래된 항목을 삭제하는 것입니다.

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

8.2 Security

일부 시나리오에서는 민감한 데이터를 처리해야 할 수 있으며 이는 솔루션 설계에 영향을 미칠 수 있습니다. 기본 수준에서 암호화는 작업 대기열에서 활성화되어야 하며 제어실 또는 세션 로그에 키 데이터가 노출되지 않도록 주의해야 합니다. 더 심각한 것은 아마도 솔루션의 어디에나 저장되지 않도록 키 데이터를 규정하는 보안 정책이 있을 수 있습니다. 이러한 요구 사항으로 인해 데이터가 전적으로 일시적이고 대기열, 로그 또는 다른 곳에서 지속되지 않도록 솔루션을 설계해야 할 수 있습니다. 예를 들어 신용 카드 번호는 '적시에' 가져와야 할 수 있으며 케이스가 처리된 후 흔적이 남지 않도록 세심한 주의를 기울여야 합니다.

9. Efficiency

9.1 Integration Efficiency

애플리케이션 통합에 대체 옵션이 있는 경우가 많으며 다양한 기술의 장단점을 고려해야 합니다. 애플리케이션에 사용자 인터페이스를 모델링하는 대신 사용할 수 있는 API 또는 웹 서비스가 있을 수 있습니다. 또는 파일을 사용자가 열지 않고도 직접 읽을 수 있습니다.

자연스러운 경향은 수동 단계를 충실하게 복제하는 것이지만 Blue Prism 이 인간이 사용할 수 없는 메커니즘을 활용할 수 있는 경우가 있을 수 있습니다. 기본 수준에서 로봇은 메모리에 더 많은 정보를 저장할 수 있으므로 사람이 두 응용 프로그램에서 정보를 수집하기 위해 앞뒤로 이동해야 하는 경우 로봇은 필요한 모든 것을 한 번에 읽을 수 있습니다.

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

9.2 Exception Efficiency

자동화 솔루션은 기술적인 이유로 또는 범위를 벗어났기 때문에 자동으로 해결할 수 없는 사례를 예상해야 합니다. 디자인은 또한 이러한 사례의 수동 해결이 어떤 식으로든 도움이 될 수 있는지 여부를 고려해야 합니다. 예를 들어 대기열 항목 상태 필드를 사용하여 다른 예외 범주를 나타낼 수 있습니다. 또는 사례 작업 중에 읽은 시스템 데이터가 정기적으로 대기열에 다시 저장되는 경우 이 추가 정보를 예외 보고서에 포함하여 수동 작업자가 사례 데이터를 다시 수집하기 위해 시스템을 탐색할 필요가 없도록 지원할 수 있습니다.

예외가 발생한 후에도 계속 작업 할 가치가 있는 상황이 있을 수 있습니다. 데이터를 가져와서 후속 수동 참조 작업을 최소화하거나 궁극적으로 예외로 표시하기 전에 케이스 작업을 계속할 수도 있습니다.

상위/하위(즉, 중첩된) 케이스가 사용되는 경우, 예외 후에도 하위 케이스를 계속 작업하는 것이 더 효율적일 수 있습니다. 하위 항목의 90%가 작업한 상태에서는 즉시 중지하는 것보다 상위 케이스를 예외로 표시하는 것이 더 낫다는 이론적 근거가 있습니다.

9.3 License Utilisation

여러 프로세스로 구성된 솔루션을 설계할 때 라이센스 사용도 고려해야 합니다. 별도의 프로세스가 최적의 구성일 수 있지만 라이센스가 프리미엄인 경우 단일 프로세스(하위 프로세스, 환경 잠금 또는 여러 대기열 사용)를 사용할 수 있는지 여부를 고려해야 합니다.

10. Notification

클라이언트의 요구 사항은 특정 이벤트에 대해 경고해야 한다고 규정할 수 있으며, 따라서 설계는 이러한 알림이 언제, 어디서, 어떻게 발행되는지에 대한 규정을 만들어야 합니다.

예를 들어, Control Room 팀은 프로세스 종료와 같은 주요 이벤트에 대한 알림을 받을 수 있습니다. 자동화가 완료된 위치에서 선택할 수 있도록 프로세스 완료를 작업에 경고해야 합니다. 또는 더 심각한 것은 비상 계획을 실행하기 위해 SLA 위반에 대해 경고해야 할 수도 있습니다.

이유가 무엇이든, 솔루션이 의사 소통하는 방법은 설계의 일부로 고려해야 합니다. 일반적으로 이메일이 선호되는 방법이지만 헬프 데스크로 보내는 SNMP 메시지도 사용할 수 있습니다. 또는 워크플로 도구 또는 MI 데이터베이스와 같은 다른 애플리케이션에 대한 업데이트로 알림을 발행할 수 있습니다.

11. Design Procedure

Blue Prism 솔루션을 프로덕션 환경에 제공하는 것은 많은 IT 프로젝트에 익숙한 Define-Design-Build-Test-Implement 의 표준 경로를 따릅니다. Blue Prism 은 문서 템플릿과 함께 입증된 설계 방법론을 제공하며, 교육생은 Lifecycle Orientation 모듈의 일부로 이에 대해 소개합니다. 그러나 이러한 템플릿은 규범적이지 않으며 일부 클라이언트는 RPA 에서 작동하도록 자체 방법론과 문서를 보다 편안하게 조정합니다. 중요한 것은 문서의 목적이며 문서 자체는 요구 사항을 정의하고 세부 사항을 노출하며 합의에 도달하기 위한 수단일 뿐입니다.

11.1 'As is' Definition and Requirements

이 문서의 다른 부분에서 언급했듯이 PDD(Process Definition Document)는 '있는 그대로' 수동 프로세스를 설명하는 데 사용됩니다. 이 정보는 SOP(Standard Operating Procedures) 또는 기타 프로세스 문서에 이미 있을 수 있지만, 생각하지 않는 로봇이 프로세스를 수행하도록 지시할 수 있을 정도로 충분히 상세해야 합니다.

수동 프로세스의 정의와 함께 자동화 솔루션에 대한 프로세스 소유자의 요구 사항을 문서화해야 합니다. FRQ(Functional Requirements Questionnaire)는 자동화된 솔루션이 사례를 작동하는 방식을 설명하는 데 사용되는 것이 아니라 프로세스 소유자와 인터뷰하여 솔루션 작동 방식에 대한 높은 수준의 요구 사항을 추출하는 수단입니다. 예를 들어 월요일부터 금요일까지 업무 시간에 실행해야 하는 지, 아니면 다른 일정에 따라 예외를 전달해야 하는 지?

11.2 'To be' Design

PDD 및 FRQ 가 완료되면 설계자는 정보를 솔루션 설계 문서(SDD)로 변환할 수 있습니다. 우리는 'as is'와 'to be'를 서로 다른 문서로 분리하는 것이 오해와 누락에 대한 훌륭한 방어책이라는 것을 발견했습니다. PDD 와 마찬가지로 워크숍 시나리오에서 SDD 를 살펴보는 것도 제안을 설명할 뿐만 아니라 단점을 노출하는 좋은 방법입니다.

SDD 는 또한 개발 책임자에게 제안된 솔루션의 품질을 평가하고 기존 로직(즉, 객체 라이브러리)을 완전히 사용하고 있으며 귀중한 프로젝트 시간이 낭비되지 않는지 확인할 수 있는 기회를 제공합니다. 디자인이 승인되면 PDI(Process Design Instruction) 및 ODI(Object Design Instruction)를 통해 개발을 시작할 수 있습니다. 이 두 문서는 디자이너가 개발자에게 빌드 방법과 내용을 알려주는 수단입니다.

11.3 Shock Proof Design

설계는 이론적인 '충격 테스트'로 테스트해야 합니다. 이것이 의미하는 바는 네트워크 중단과 같은 갑작스러운 타격에 솔루션이 어떻게 대응할 지 상상하는 것입니다.

프로세스가 케이스 처리 중에 있고 모든 네트워크 통신이 끊어지고 데이터베이스 연결이 끊어지고 프로세스가 종료된다고 상상해 보십시오. 네트워크가 복원되고 프로세스가 다시 시작되면 어떻게 됩니까? 이전 케이스는 어떻게 되나요? 프로세스가 다시 작동하면 단계가 중복되거나 중단된 부분에서 계속 진행됩니까? 응용 프로그램이 열려 있는 경우 프로세스에서 처리할 수 있습니까? 케이스 중간에 갑작스러운 이벤트가 발생하는 것 외에도 대기열을 로드하는 동안 발생한 경우 중복이 추가되거나 충격이 완화되면 케이스가 손실됩니까?

그러한 상상은 비관적으로 보일 수 있지만 오작동하는 솔루션의 결과가 충분히 심각한 경우 필요할 수 있습니다.

11.4 Application Assessment

새로운 대상 시스템이 범위에 들어올 때마다 애플리케이션 평가를 수행하는 것은 항상 가치가 있습니다. 이를 위해서는 자동화할 새 시스템의 요소에 대해 Application Modeller 스파이를 실행하여 얼마나 쉽게 식별할 수 있는지 확인해야 합니다. 일반적으로 Windows, 브라우저 및 Java 애플리케이션의 경우 이 활동을 수행하려면 개발자가 각 요소 유형(콤보 상자, 확인란, 테이블, 목록 등)과 상호 작용하는데 가장 적합한 기술을 결정해야 합니다. 이는 개발 노력을 추정하고 케이스 처리 시간의 초기 근사치를 제공하는데 도움이 됩니다.

11.5 Project Types

Proof of Concept

POC 의 목표는 무인 자동화가 가능한 완전 자율 솔루션을 생성하는 것보다 자동화의 잠재력을 입증하는 것이어야 합니다. 이를 위해 설계는 확장성, 애플리케이션 제어, 예외 처리, 알림 및 MI 와 같은 프로덕션 솔루션에 필요한 기능보다는 범위 내 사례 작업에 더 중점을 두어야 합니다.

분명히 솔루션이 작동해야 하지만, 완전한 생산 제품일 필요는 없으며 데모 목적으로 참석 모드에서 주로 제한된 기간 동안만 실행될 가능성이 높습니다. 디자인은 가능한 시간에 현실적으로 전달할 수 있는 것을 고려하고 프로젝트의 전반적인 목적이 무엇인지 기억해야 합니다. 개념 증명에 따라 설계 검토를 수행하고 최종 설계를 문서화하고 추가 개발을 통합하여 프로세스가 생산 강도를 유지하고 공식 테스트 준비가 되었는지 확인해야 합니다.

Pilot

파일럿은 프로덕션 사례에 대해 일시적으로 실행되도록 빌드된 프로세스입니다. POC 와 유사하게 파일럿 프로젝트의 설계는 프로젝트의 목표를 반영해야 합니다. 파일럿의 목적은 완전한 솔루션을 만드는 것이 아닐 수 있으므로 솔루션이 무엇을 할 것인지, 하지 않을 것인지에 대해 합의된 범위를 설계에 포함해야 합니다.

파일럿이 이전 POC 를 기반으로 하는 경우 원래 설계를 재검토하도록 주의해야 합니다. 위에서 언급했듯이 편의를 위해 POC 는 솔루션의 견고성을 제한할 가능성이 높으며 후속 파일럿에서 개선해야 합니다.

11.6 Design Authority

디자인 책임자의 주요 활동은 다음을 정의하고 관리하는 것입니다.

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

설계 책임자는 다음과 같은 일반적인 기술을 사용하는 방법을 결정해야 합니다.

- Email–email clients or SMTP?
- 워크플로 시스템 – Blue Prism 은 기존 워크 플로우 시스템을 어떻게 활용하여 수동 사용자와 상호 작용합니까?
- 제 3 자 코드-어떻게 래핑되고 Blue Prism 에 노출되어야 하는가.
- Web services

디자인 책임자는, 예를 들면 프로세스, 객체, 작업 대기열, 환경 변수, 데이터 항목 등에 대한, 디자인 및 개발 시 명명 규칙을 정의해야 하며, 로깅 정책 및 데이터 저장 정책을 규정하고, 그리고 개발 모범 사례의 관리자가 되어야 합니다.

Appendix

12. Design Review Checklist

12.1 Solution

- 'as is' 수동 프로세스를 정의하고 자동화 솔루션을 설계하기에 충분한 세부 정보로 문서화했습니까?
- 고객의 요구 사항이 문서화되고 동의됩니까?
- 이 설계에서 요구 사항을 충족합니까?
- 설계를 통해 고객이 'to be' 자동화 솔루션을 이해하고 승인할 수 있습니까?

12.2 Process

- '사례 작업' 단계 외부에는 어떤 논리가 있습니까?
  - '준비' 단계가 필요합니까?
  - '최종화' 단계가 필요합니까?
  - '재설정' 단계가 필요합니까?
  - '복구' 단계가 필요합니까?

- 프로세스가 Blue Prism 작업 대기열을 사용합니까?
  - If not, why?

- 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 사용되는 암호는 어디에 저장합니까?
  - Credential Manager 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 시정될 것입니까?
    - 응용 프로그램을 다시 시작해야 합니까?

- 예외는 어떻게 관리됩니까?
    - 대기열 재시도가 사용됩니까?
    - 예외는 어떻게 비즈니스에 전송됩니까?
    - 예외율은 어떻게 모니터링됩니까?

- 대기열 결과를 다른 Blue Prism 대기열, 워크플로 응용 프로그램, 데이터베이스 또는 파일에 복제해야 합니까?
    - 언제, 어떻게 완료됩니까?
    - 양면이 불균형해질 가능성이 있습니까?

- 알림을 보내는 데 필요한 프로세스입니까?
    - 이 작업은 언제 어떻게 수행됩니까?

- 하위 프로세스를 사용합니까?
    - 메모리 누수의 위험이 있습니까?

12.3 Objects

- 이 솔루션에 필요한 개체가 항목별로 분류되었습니까?
  - 이미 존재하는 객체와 생성해야 하는 객체가 명확합니까?
  - 개발 작업을 시작하기 전에 설계를 검토할 수 있는 체크포인트가 마련되어 있습니까?

- 이 솔루션을 위해 생성된 객체를 재사용할 수 있습니까?
  - If not, why?

- 재사용을 불가능하게 할 수 있는 '비즈니스 프로세스 로직'이 객체 계층에 존재합니까?
  - 이 로직이 프로세스 레이어에 있어야 합니까?

- 시스템 예외 외에, 프로세스에서 특별한 처리가 필요한 예외를 throw 하는 객체가 있습니까?
  - 이것은 객체의 사용자에게 명확합니까?
  - 출력 매개 변수가 더 명시적입니까?

- 나눌 수 있을 만큼 지나치게 복잡한 페이지가 있습니까?
  - For ease of use
  - For ease of reuse
  - For more effective testing
  - For increased efficiency
  - For better security

- 객체 및 페이지의 이름이 프로세스 개발자에게 목적에 대한 좋은 아이디어를 제공합니까?
  - 의미가 없거나 모호한 이름이 있습니까?