

Contents

Exam Topic Percentages.....	3
PRPC Overview	3
Developing a PRPC Application.....	4
Project Initiation	4
Suggested Activities	4
Key Deliverables.....	5
Additional Information	5
Inception.....	5
Suggested Activities	5
Key Deliverables.....	5
Additional Information	6
Elaboration	6
Suggested Activities	6
Key Deliverables.....	6
Additional Information	6
Construction	7
Suggested Activities	7
Key Deliverables.....	7
Additional Information	7
Transition.....	7
Suggested Activities	7
Key Deliverables.....	8
Additional Information	8
Go Live.....	8
Suggested Activities	8
Key Deliverables.....	8
Direct Capture of Objectives	10
Gathering and Entering Requirements	13
Flow Design	14
UI Design	18
Data Modeling	21
Automating Business Policies	24
Creating Messages and Notifications.....	26

Building Reports.....27

Business Architect Best Practices28

 How to build for change33

Exam Topic Percentages

Test Competencies	% of Exam
PRPC Overview	4.29%
Developing a PRPC Application	14.29%
Gathering and Entering Requirements	20%
Flow Design	15.71%
UI Design	15.71%
Data Modeling	4.29%
Automating Business Policies	10%
Creating Messages and Notifications	4.29%
Building Reports	5.71%
Business Architect Best Practices	5.71%
Total	100%

PRPC Overview

■ Navigating the Business Analyst portal

■ Routing and assignments

- Understand the purpose of – and how to create – worklists and workbaskets
 - Workbasket
 - A shared queue of work items from which users can retrieve work – ordered by urgency
 - Assignment can leave a workbasket as a result of:
 - Removal by a qualified user – i.e. user specific to the organisation hierarchy and work group or assigned this WB via their Operator ID
 - Manager re-allocating the work item to another workbasket or worklist that they manage
 - Automated agent can route to another a user base on work schedule, skills, due date, workload etc.
 - Create by Pega>Org&Sec>Work Basket, create with a recognisable name ending in WB, associate with a work group
 - Use a router to route an assignment of the workbasket – the router will call rule ToWorkbasket with the workbasket name defined as a parameter
 - Can also use ToSkilledWorkbasket or a ToDecisionTree/Table to route to the right workbasket
 - Worklist
 - These are associated with a single operator

- Work items are routed to a worklist by routing to the operator using ToWorklist in a router with the Operator ID as the parameter
- Can use CheckAvailability to check if operator is available based on calendar etc set up with the operator
- Again can use decision as part of routing
- Best practice to not hard code operator Ids

■ The 6 R's

- **Receiving** — Accepting and capturing the essential data describing work from multiple sources in multiple media and formats, from keyboards, scanners, and external systems.
- **Routing** — Using characteristics of the work and knowledge about the workforce to make intelligent matches and assignments.
- **Reporting** — Providing real-time visibility of work in progress, work completed, productivity, bottlenecks, and quality.
- **Responding** — Communicating status, requests for information, and progress to the work originator and to other people involved in the work, by email, fax, written mail, and other means.
- **Researching** — Accessing external systems and databases through connectors to support analysis and decision making.
- **Resolving** — Through automated processing and automated support of users, completing the work and updating downstream systems promptly.

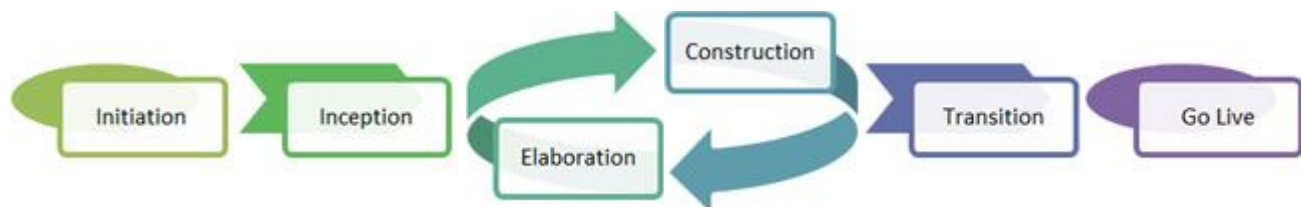
Developing a PRPC Application

■ Application development steps and stages

The Pega BPM methodology is an agile process framework and is adaptable. Pega BPM is intended to be flexible, so that the implementation team can adapt it to any sized project, and right-size the Pega BPM process to their needs. The resulting process should reflect the goals of the iterative phases. It is critical to promote business analyst functions throughout the iterative phases as part of:

- Change management processes
- Stakeholder communication and expectation setting
- Application metrics reporting, measurement and data analysis on productivity gains
- Discovery of further process improvements

Click each phase below for more information.



Project Initiation

Suggested Activities

- Preliminary discussions regarding the scope of the over-all program (or project)
- Accomplish walk-throughs of the current process; include all parties and review all applicable systems
- Determine quick wins for the client and decide which slivers are priorities
- Conduct knowledge transfer sessions and begin enablement training

Key Deliverables

- Proof of Concepts
- Workshops with demos of current state versus targeted process improvements and process re-engineering
- PRPC Enablement training courses
- Project Roadmap planning; identify critical milestones, staffing model proposals, risks and assumptions

Additional Information

[Implementation and Methodology Overview](#)

[Guided development through guardrails](#)

[Ten Guardrails to Success](#)

[Top Ten Usability Guardrails](#)

Inception

Suggested Activities

- Define the scope of the over-all program, project, or specific sliver
- Define the business requirements and organize them into projects and then logical slivers of deliverables
- Confirm the sliver and targeted deliverables
- Capture the high-level requirements, building out the use cases and work types for the sliver
- Perform operational walk-throughs to identify key process improvements with measurable targets
- Estimate the effort with the teams involved

Key Deliverables

- Inventory and analyze what artifacts already exist, identifying reuse and leveraging existing artifacts
- Initiate DCO sessions (conduct either prep & review, white board & review, or real-time capture DCO sessions)
- Enter the DCO session artifacts directly into the Application Profiler for each sliver
- As the Application Profile is built-out, use the Sizing Tool in concert to size the effort
- Review the Application Profile & and project sizing results with stakeholders
- Leverage the built-in Project Plan Template for a high-level draft of the initial project plan
- Establish the Project Management Framework (PMF) and Test Management Framework (TMF) for use with the project
- Update the Resource Plan and the Risk and Assumptions with any changes
- Complete a Phase Readiness checklist
- Perform a Methodology Alignment Workshop, if blending Pega BPM best practices and DCO with a client mandated methodology

Hold a project kick-off meeting with resulting drafts of the Application Profile, project plan, resource plan and project sizing

Additional Information

[Introduction to the Business Analyst Portal](#)

[Defining and using Key Performance Indicators in the Business Analyst portal](#)

[About the Direct Capture of Objectives](#)

[back to top](#)

Elaboration

Suggested Activities

- Complete the detailed DCO sessions (conduct either prep & review, white board & review, or real-time capture DCO sessions)
- Capture requirement details, further building out the Application Profile with atomic use cases and work types
- Work iteratively, cycling through Elaboration and Construction phases
- Build out the foundation of the proposed application completing the Application Profile and running the Application Accelerator
- Environment setup, planning and execution
- Clarify and establish a governance model if it is not already present
- Establish the Deployment Plan/ Process, the Test Plan, and Issue Tracking Process
- Use the Pre-Flight and PAL tools

Key Deliverables

- Complete the Application Profile with the atomic use cases detailed
- Produce the automated document generation from the Application Profile and update the sizing effort
- Run the Application Accelerator to establish the application foundation
- Create the draft flows and draft UI
- Establish the links between the requirements, cases and rules
- Engage PMF and TMF established in the previous phase
- Create the detailed project plan
- Develop a testing plan, migration plan and establish standards for executing the plans if they do not yet exist
- Lead a change management review to determine if requirements or priorities have changed
- Updated Deployment Plan and Test Plan
- Work iteratively and review work to-date

Additional Information

[Designer Studio Overview](#)

[DCO 6.2 - Creating Application Profiles and Discovery Maps](#)

[DCO 6.2 - Using the Application Accelerator](#)

[DCO 6.2 - Using the Application Document Wizard](#)

Construction

Suggested Activities

- Assign application configuration tasks using the iterative model
- Unit test each component
- Build test scripts for automated testing
- Run application test scripts and begin the system test process
- Complete integration system testing
- Fully use PMF and TMF established during Elaboration to track progress and triage defects
- Hold a Design review or User Experience review
- Review PMF and start preparations for the next sliver
- Lead a change management review to determine if requirements or priorities have changed
- Use the Pre-Flight and PAL tools

Key Deliverables

- Using PMF, assign the work from the DCO sessions and tasks from the project plan
- Resources complete the assigned work/ unit tests
- Build a Release Candidate meeting the requirements of the sliver
- Complete Unit/ Integration testing and associated Issue tracking Process
- Complete all iterative automated testing
- Engage TMF to test, report on and ensure rule quality
- Build and test migration plans
- Design review and user experience review with stakeholders
- End-user training, production deployment and Support guides, as needed
- Update any requirements and regenerate documentation using the Application Profile
- Performance, Pre-Flight, PAL, and log file analysis
- Updated project plan
- Complete a Phase Readiness checklist

Additional Information

[How to customize the login screen](#)

[How to customize the favicon for the User and Manager composite portals](#)

[About Automated Testing](#)

[About the Project Management Framework Version 6.2.2](#)

[back to top](#)

Transition

Suggested Activities

- Environment migrations completed for QA and UAT
- QA and UAT testing completed
- Automated testing cycles completed

- Issue Tracking process completed with TMF review
- Participate in the system testing and UAT process to ensure the application is acceptable to stakeholders and business users
- Ensure solution is maintainable by IT, including Support
- Review PMF and finalize plans for the next sliver
- Formalize stakeholder sign-offs and readiness
- Revisit target metrics and measurements for success
- End-user training, rollout and support plans completed

Key Deliverables

- Participate in Go/ No-Go status update review after formalized stakeholder sign-offs
- Migration Delivery documents and production roll-out run book completed
- Production Support plan and Support training completed
- End-user training completed
- Issue tracking and resolutions should be signed-off
- Review reports, what metrics will be gathered and how follow-up will occur
- Completed QA/automated testing and UAT testing
- Performance, Pre-Flight, PAL, and log file analysis should be signed-off
- Complete a Phase Readiness checklist

Additional Information

[Using the Report Viewer to define and evolve simple reports](#)

[Using the Report Browser to Organize and Manage Reports](#)

[About Standard Management Reports](#)

[Understanding Monitor Processes reports in the Monitor Activity workspace](#)

[Understanding Monitor Assignments reports in the Monitor Activity workspace](#)

[Understanding Analyze Performance reports in the Monitor Activity workspace](#)

[Understanding Analyze Quality reports in the Monitor Activity workspace](#)

[Preparing to create and use interactive charts](#)

Go Live

Suggested Activities

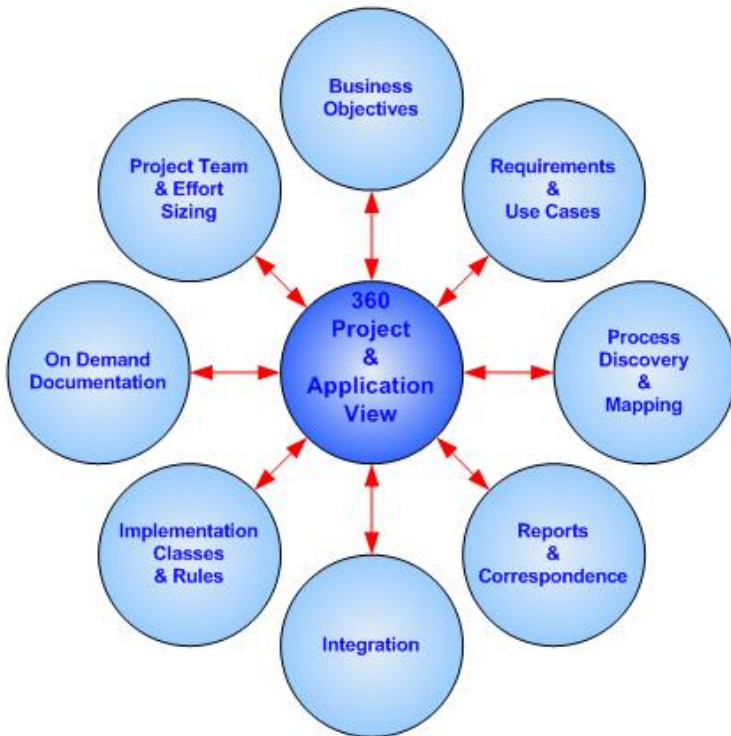
- Data migrated into Production
- Application deployment
- Go-Live support activities

Key Deliverables

- Deploy final release candidate
- Maintain application with enhancements, fixes, and change requests by production support and business users
- Production Support/ Documentation updates, as needed
- Monitor system using reporting tools, identify and capture process improvements and related requirements

Plan and deliver a status update to all stakeholders after an established period for metrics to be adequately reported on

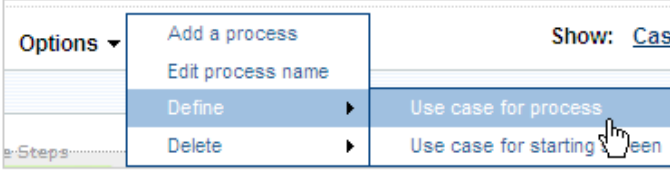
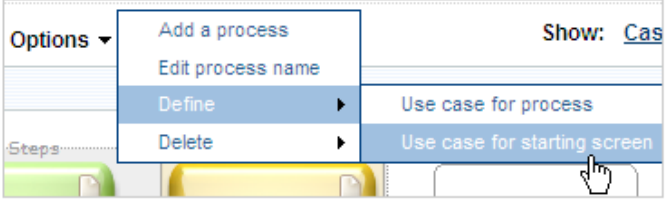
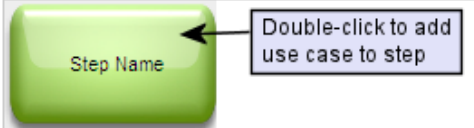
- Direct Capture of Objectives - fosters a Build for Change mindset that encourages reuse of application assets, automates common project activities, and generates on-demand, up to date, accurate project documents (DCO). The end result and benefit of using DCO is a clear 360 degree view of your Process Commander application through a project life-cycle.



Direct Capture tools include:

- **Application Profiler** — captures objectives, use cases, requirements, processes, interfaces, reports, correspondence, and case type relationships; estimates project sizings; documents the collected information as an application profile document; jump starts application development by providing the information as input to the Application Accelerator.
- **Application Accelerator** — jump starts the creation and extension of your application and automates best practices for application design; consuming information from an application profile, it creates organization and class structures, case type relationships, and draft processes and user interface elements so you can see the working application based on the captured information.
- **Application Document wizard** — supports rapid and iterative documentation of the application as development progresses using your predefined templates and content settings.
- **Application Use Case** — allows business users to describe at an atomic, granular level the steps required to build an application in business language; stored as business rules and linked to processes and other application components to provide development and testing detail and traceability as development progresses.
- **Application Requirement** — allows business users to describe capabilities that the application must fulfill and define success criteria; stored as business rules and linked to other rules to provide development and testing detail and traceability.

When using the Application Profiler to create an application profile, you can specify use cases and requirements either on the **Create Processes** step or the **Project Explorer** step.

For	How	Example
The overall process	Next to the Starting Process drop-down list that displays the name of the process, select <i>Options > Define > Use case for process</i> .	
The process's starting screen	Next to the Starting Process drop-down list that displays the name of the process, select <i>Options > Define > Use case for starting screen</i> .	
A step in the map	Double-click the step (shape)	

■ Business Architect responsibilities

■ Rules and Classes

Direct Capture of Objectives

- Phases of the development methodology:
 - Name the project phases:
 - Inception
 - Elaboration
 - Construction
 - Transition
 - Describe the goals and deliverables of each phase:
 - Inception
 - To define the scope of the project by understanding the business requirements to allow the definition of a project and slivers. Construction of the AP object and proposal
 - Elaboration
 - To build out the foundation for the project, perform DCO workshops to extract detail for the requirements, use cases and work types, to define draft flows, draft UI, develop testing plan, establish standards
 - Construction
 - Take the output of the Elaboration phase, split into iterations (if possible), assign work, configure the application, unit test and begin system test
 - Transition
 - Finish testing, build application and deliver into production
- Directly capturing objectives:
 - Describe the inputs to the process
 - The Application Profile wizard is used to capture the information required for this phase:

- Steps:
 - Processes
 - Interfaces
 - Reports
 - Correspondence
 - Assumptions
 - Project Roles
 - Review
 - Actors
 - Generate
- Capture basic project details:
 - Application name
 - Project name
 - Organisation
 - Business objectives
 - Description
 - Actors – can be human, PRPC or external system
- Processes:
 - Capture steps of the business process for each work type
 - Subflows
 - Human actions
 - Auto actions
 - Intergration points
 - Represented as rectangles
 - Map primary flow first
 - Map subflows below primary flow step
 - Map alternate flows
 - Use cases can be added to each process
 - Business requirements can be associated with use cases
 - All items (processes, reports, integrations, correspondence) will have complexity associated with them – this is used to generate the effort estimate for the project
- Integration
 - Captures the systems that PRPC will integrate with
 - Capture basic info on these systems
- Reports
 - Capture the reports that will be produced
- Correspondence
 - Capture the items of correspondence that will be generated and describe
- Assumptions
 - Capture any project specific assumptions
 - Lots of boiler plate assumptions
- Project Roles
 - Capture the roles of the project team
 - This is used with the effort estimate to work out the project duration, broken down by phase

- Both internal and external
 - Actors (not in the main flow)
 - Can set option for just new framework, new implementation or both
- The AP wizard generates an application profile and a proposal document, creates the organisation name (e.g. iLend)
 - The application profile is consumed by the AA (Application Accelerator)
 - The proposal document is used to get sign-off for the project from the client
- The Application Accelerator:
 - Steps:
 - Base and Rulesets
 - Processes
 - Class structure
 - Reports
 - Correspondence
 - Role (application roles)
 - Actors
 - Review
 - Generate
 - Base and Rulesets
 - Captures the organisation structure used to generate the ruleset names and class structures
 - Organisation
 - Division
 - Business Unit
 - FW = Org-FW-Framework nameFW-
 - Impmentation = Org-Division-Application
 - Processes
 - Review the processes as specified in the AP
 - Class structure
 - Define the class structure under the Work- class
 - Check the store separately check box to have the class stored in separate DB table (by default all the classes will be stored in the same DB)
 - Can add layers to promote specialisation and increase re-use
 - Can set direct inheritance in this step
 - Reports
 - Review from AP and add any further detail
 - Correspondence
 - Review from AP and add any further detail
 - Roles
 - Capture details on the party roles that are going to receive correspondence
 - Actors
 - Review and add to any actor info captured in the AP
 - Review
 - Can review all the rules in a big list, ordered by type

- Generate
 - Generates:
 - Use case/requirements docs
 - Sets up the initial application structure
 - Class structure
 - Starter flows based on high level processes
 - Sub flows based on high level processes
 - User interface rules
- 10 [Guardrails](#)
 - Iterative approach
 - Robust foundation – good class structure
 - Nothing that is hard
 - Limit java customisation
 - Design for change
 - Regular performance monitoring
 - Intent driven design
 - Easy to read flows
 - Declarative NOT procedural
 - Security is OO too

Gathering and Entering Requirements

[PDN dco-62-and-63-working-with-application-use-cases-and-requirements](#)

Application use cases describe, using business language, the steps required to process and resolve work in an application. You associate use cases with a work type's processes, subprocesses, and the individual process steps.

Usually, you specify application use cases when you [create the application profile and discovery maps](#) for the application using the Application Profiler. For a given discovery map, you can associate a specified use case with:

- Individual steps in the process, including any subprocesses
- The starting screen of the process
- The overall process

Those use cases specified at the process and subprocess level are typically described as *comprehensive* use cases. Like the traditional UML use case, such use cases usually describe the business process for a single work type from beginning to end.

Those use cases specified at the level of the starting screen or individual process steps are typically referred to as *atomic* use cases. These describe the smaller, more granular steps taken within the comprehensive use case. Because they are atomic, they can be reused to ensure consistency across other processes in an application, and make it easier to update processing steps when you need to implement a change.

Atomic use cases:

- Specify one or more actors
- Specify a single event or method that triggers it; think of it as one short process in a comprehensive use case
- Do not involve a change of ownership during the steps of the atomic use case
- Correspond to one particular step or series of steps within a screen flow type of process, or within a single flow action in the processing of a work type

- Describe the processes to be performed including the steps involved in completing the use case, applicable edits, and the expected behaviors and outcomes
- Provide enough business detail so that a developer can implement it and a user can test it
- Take only a few minutes to enter

■ Collecting and storing requirements in PRPC

■ The relationship between requirements and specifications

Application requirements describe, using business language, the project and processing needs that must be met for processing work and data in the application. Think of requirements as an inventory of events, conditions, or functions that must be satisfied and tracked in a development project. They also provide benchmarks against which the application can be tested. Typically, you associate requirements with a corresponding use case, which itself is associated with the relevant processes, subprocesses, and process steps.

A good requirement:

- Describes the need in business terms so that it is easily understood by the implementation team
- Has an identified category type (such as "Enterprise standard" or "Business rule") and level of importance to the application
- Indicates its current implementation state (such as Open or Withdrawn)

■ The Application Profiler, Application Accelerator, and application profiles

[see above]

■ Work types and case management

■ DCO Elaboration sessions

Flow Design

■ Using the Process Modeler

Assignment — A point in the business process that requires human judgment and input. Flow processing of a work item normally pauses when it reaches an assignment shape until a user completes the assignment. The assignment normally appears on the worklist of the user who starts the flow.

Flow — A flow rule is the fundamental representation of a business process in Process Commander. The rule defines the sequence of processing that your application applies to work objects. It contains a network of shapes and connectors. They are associated with rules, parameters, or property values that precisely specify how the shapes and connectors act upon the work object.

Straight-through processing — Business processes that can usually or often be performed completely automatically, with no human intervention. A flow rule that contains no assignment tasks by definition implements

straight through processing. This represents a process with the greatest potential to maximize efficiency and return on investment.

Screen flow — A flow rule that presents a user with a sequence of forms to complete. A screen flow provides an effective way to simplify input processing and present questions or input fields in a series of related forms. Users can change an answer to an earlier question by backing up in the flow; and, in some situations, can complete steps in any order.

Shapes — Each shape in Process Modeler represents an Assignment, a Utility, a Decision, a Start or End, and so on. (The Microsoft Visio presentation of a flow are informally known as a tasks.)

Work item — The primary unit of work completed by an application, and the primary collection of data that a flow operates on. Operators using an application create, update, and eventually resolve and close work items.

Worklist — A list of outstanding (not complete) assignments waiting for a user to perform them. The list appears in a portal that supports application users that create, update, route, and resolve work objects.

Workbasket — A queue of open assignments that are not associated with an individual operator. Managers can review the assignments placed in a workbasket and distribute them one-by-one to appropriate users for processing. Alternatively, the system can automatically reassign assignments from the workbasket periodically to a user's worklist.

■ Flow rules

- These shapes are linked to together to form the flow
- Assignments are assigned to an operator/worklist/workbasket
- Flows can be added as sub-flows
- Flows can be added as screen flows
- Connectors in a flow are associated with flow actions that are associated with sections and can call activities
- Utilities can call activities
- Integrators call connectors to link to external systems
- The flow can be set to draft so that it can still be run even if not all shapes have everything completed
- Should restrict the number of shapes (excluding connectors, notifiers and routers) to 15 or less (guard rail)
- Create and modify flow diagrams:
- Some flows are created out of the AA (starter flows for the work types)
- Other flows can be created by dragging a flow shape onto a flow (for a flow or screen flow)
- Can be created through right clicking process in the correct class and clicking new flow

■ Sub flows and screen flows

- Design and implement a screen flow
 - Create a new flow of type ScreenFlowYYY to create the flow
 - The screen flow is then edited in a similar way to a normal flow, however the flowactions are on assignments not connectors
- Flow actions
 - Define and create flow action rules

- Flow actions are created by right clicking the UI category in the relevant class
- Also created by running the flow in draft mode – the system will prompt the user to create a flow action
- A flow action is a UI component that references a section and is used to call that section –an assignment can have multiple flow actions associated with it, they will be displayed as options in the UI ordered by the probability associated with the flow action
- Flow actions can call activities
- Flow actions can be used to enforce validation on properties by calling a validation rule
- Describe various methods to define HTML in a flow action:
 - The HTML can be customised by this is a bad idea
 - Sections can be added to the flow action – this will have the HTML generated automatically
- Call activities before and after a flow action executes
- Understand the difference between connector flow and local flow
 - Connector flows progress the work item through the flow – the is an update Status action that can be called as part of this
 - Local flows just do something to the object without progressing it
- Understand how to use and modify standard flow actions
 - Think this is standard harnesses
 - There are four standard harness rules
 - New – creates the work item
 - Perform – calls a flow action
 - Review – present work item as read only
 - Confirm – acknowledges completion of an assignment with a confirmation display of the work item

○

■ Flow shapes and their uses

Smart shapes are:

- Start Flows
- Flows
- Connectors
- Assignments
- Utilities
- Routers
- Notify
- Decisions
- Forks
- Integrators
- Swim lanes
- Split-join
- Spin off
- Split

■ Work item status

- Work status
 - Describe the standard work statuses and how they are categorized:
 - Changes as a work item progresses through a flow
 - There are four standard work statuses
 - New
 - Open
 - Pending
 - Resolved (resolved-rejected and resolved-completed)
 - Custom work statuses can be created but they must always be prefixed with one of the standard work statuses
 - Work status is set on entry to an assignment, by utilities and by activities
 - pyStatus holds the work status
 - Apply work statuses to work objects
 - This is set:
 - On entry to an assignment
 - By a utility
 - By an activity
 - Understand how work is resolved
 - The status is set to resolved (completed or rejected)
 - It requires no more processing
 - The work object is kept in the DB
 - It does not appear in any worklists or workbaskets
 - It has to be re-opened for any further changes to be applied to the work item
- Service Levels
 - Understand the time intervals on which service levels operate
 - There are 3 milestones
 - Goal – the time by which an assignment should be completed
 - Deadline – the time within which an assignment must be completed
 - Past deadline – the assignment is past the deadline
 - The time for each milestone is set in the Service Level milestone associated with the assignment
 - Goal is days/time from start of assignment
 - Deadline is days/time from start of assignment
 - Passed deadline is days/time from deadline milestone
 - The SLA is associated with the assignment through the assignment properties panel
 - Execute an activity based on reaching a goal or a deadline????
 - The SLA is monitored by a system agent
 - If a milestone is reached then an activity can be kicked off – the activity is specified in the Escalation Activity field on the SLA (along with any parameters) – typically used to push the assignment to a manager or to a workbasket
 - Understand the concept of urgency as it relates to the work object and an assignment
 - OOTB all assignments are given an urgency of 10
 - A specified amount is added when the assignment begins – this is specified in the SLA
 - A further amount is added when each of the milestones is reached
 - Worklists and workbaskets can be ordered by urgency
 - Work can be automatically assigned to a worklist by urgency

UI Design

■ Basic UI rules and their uses

– Harness

– Flow Action

– Section

■ UI wireframes

■ Layouts and their uses

■ UI Controls

- User Interface Rules

- Define how different HTML rule types are used to construct the user interface:

- Harness

- The harness is the highest level container for HTML rule types

- Defines the top level structure of a work item form

- Calls section rules

- There are 4 standard harnesses

- New

- First harness called in a start flow

- Will call the first section rule

- Creates the work item

- Perform

- Allows users to select a flow action, presents the flow action

- Review

- Presents the work item in a read only mode

- Confirm

- Acknowledges completion of an assignment with a confirmation display of the work item

- Standard harness contains

- Header

- Body – usually put containers/sections in here

- Footer

- The AA creates the New, Perform and Confirm harnesses

- To edit/view the harness

- Select the start shape in the flow editor

- Right click the flow shape in the diagram tab of the flow rule

- The new harness is reference on the process tab of the start process

- There are also screen flow harnesses

- PerformScreenFlow

- TabbedScreenFlow

- TreeNavigation

- Flow actions in a screen flow display within one harness

- Section

- Contain embedded UI elements such as labels, fields, forms, controls, properties etc
 - Can contain other sections
 - Are referenced by flow actions and harnesses
 - Can be re-used by multiple HTML rules
- Flow Action
 - References section(s) when an action is selected
 - The flow actions from an assignment are given percentage figures – these are used to rank in the action list
 - The flow action with the highest percentage is the default and will display it's associated section
 - Flow actions can call activities as pre or post processing steps
 - Flow actions can call validation rules to validate section properties when the flow action is called
- HTML Property
 - These format the elements in a section – control a property's appearance for both input and display
 - Can be customised
 - Can have a number of parameters – accessed through the magnifying glass icon next to Display
 - Autocomplete is an HTML property
 - Dynamic select is an HTML property
- Identify the standard harnesses and their functions
 - New
 - Perform
 - Review
 - Confirm
- Describe how sections can be embedded in other HTML types
 - Can be embedded in Harnesses, Flow Actions and Sections
- Use the Rules Inspector to identify HTML rule types and properties
 - If this option is switched on then an H will appear in the UI for all HTML rule types
 - Clicking on the H opens the rule
- Understand the benefit of automatically generating HTML and the consequences of overriding
 - The benefit is that HTML can be created quickly, with constant behaviour and it is easy to maintain
 - Manually creating HTML means that the standard tools used to create the HTML rules will no longer work, this makes maintenance harder especially when performing an upgrade. These customisations will have to be maintained locally in the implementation and will not benefit from improvements delivered by Pega in new versions
- Property Display
 - Use HTML property rules to define field display and input formats for calendars, text fields and select lists
 - These are not HTML property rules
 - Can add sections, containers, tabs, layouts
 - Layouts – can be:
 - single, double, triple
 - Repeating – row, grid, tree, treegrid

- Grid is a simple table with repeating rows
 - Tree is like file explorer
 - TreeGrid is a combination with a tree that opens to a grid
 - Freeform
 - Like setting up tables in word
- HTML property rules are used to link the field to a property in the class structure
- Define the property that the field references – this is what will be displayed/updated
- Define how it will be displayed – for some more complex display types this may require an activity to be called to populate the list, if nothing is selected the display format will be taken from the property
- Behaviour – this defines what happens when ONCLICK,ONCHANGE,ONBLUR
- The fields can be made mandatory – an orange asterisk appears next to the label and the field must have a value entered
- Allow client side dynamic display functionality
 - Client event
 - Renders HTML when a user performs an actions
 - Will need to refresh screen to display new clip board data or conditionally visible fields
 - Conditionally display fields
 - Use a When rule to determine when they are visible
 - Dynamic selects
 - Displays a list of choices returned from an activity or a list view
 - Can link dynamic selects to refine lists as users make choices
 - Dynamic calculations
 - Can perform calculations on data entered by the user to be displayed on the UI
 - Smart pop ups
 - SmartInfo – displays a lightweight CSS pop up on hover over
 - Used to provide additional detail
 - Can display as embedded for reports only
 - It can be clickable
 - A sections must be defined for the pop up
 - Smart label
 - Smart info for static labels
- Use Edit Input rules to convert data???
- Validation
 - Understand how and when validation rules are used
 - Several types of validation
 - Making the field mandatory – check the required box in the field properties
 - Property rule validation
 - Defined on the property
 - Can be against a format or a local list
 - Create a Validate rule
 - These are called from the flow action
 - The rules are defined in the process category
 - They validate the data entered in the section called by the UI
 - Can also be called by activities

- Can validate:
 - Single properties – validate
 - List – validate each
 - Can call another Validate rule for an embedded page
 - Have a test against a single value
 - Check boxes on Required, Conditions (allows detailed conditions), Continue (process other rows even if the rule fails on this row)
 - Many expression types can be selected
- If validation fails then a red X is placed over the field
- A message will pop up if the user hovers over the red X – the message can be defined
- Better to use declaratives on the properties
- Portals
 - Standard portals are:
 - User – user
 - Manager – manager
 - Designer – business architect, system architect (these two can use the other portals as well)
 - User portal allows a user to create, update and resolve work items
 - Manager portal allows
 - Monitor work item creation
 - Review service level compliance
 - Run reports
 - Modify business rules and process relationships
 - Describe how to associate users and their portals
 - Users are associated to their portals through access groups – will be covered in security
- List view rule
 - Provides an HTML report of class instances
 - The class applied to determines the class to report on
 - Has its own type of form to enter data:
 - On Content tab:
 - Has criteria to select records – condition on class property and value (can be a param passed to the list view)
 - Select the fields to return to the clipboard page
 - Define clipboard page to hold results
 - On Display fields tab
 - Property fields and labels to display in the list
 - Embedded list view is contained within a cell in a layout

Data Modeling

- Using properties to store data in PRPC applications
- Property types available in PRPC
- Property modes available in PRPC
- Standard PRPC properties

- Data Classes
 - Understand how and where application data is stored
 - Application data is stored in properties associated with a work item
 - Properties are stored in the DB in a BLOB with the work item
 - The work item, along with data is pulled into memory to be read, updated
 - Data objects inherit from Data-, either directly themselves or by inheriting from a class that directly inherits from data-
 - Data objects are not parts of class groups or class groups
 - Data classes must be concrete
 - Create a data branch in an application
 - New Rule, use RuleSet name, e.g. iLend-FW-LendFW would mean creating a data class of iLend-FW-LendFW-Data-<Class name>
 - Must be concrete and not belong to a class group
 - Use data classes to store information related to work
 - The data class must be referenced by the properties on a work item
 - Use data classes as the basis of embedded pages, page lists and page groups within a work object
 - To create an embedded page create a data class containing the data items that you want to have in the embedded page
 - Reference it by adding a property of type Page to the work type
 - For a page list or page group do the same but add the relevant property type
- Properties
 - Identify the correct class in which a property should be defined
 - The property should be defined against a class that maximises re-use, generally at the highest level at which all child classes use the data
 - This applies both to the data and work classes that reference them
 - All properties that apply to the same thing should be in the same data class to allow them to be embedded as a page/page list/page group
 - Identify and describe the standard property prefixes
 - px – these are computed properties that we can see but not update
 - py – these are used for OOTB PRPC properties that are inherited from PRPC classes, these can be used in the FW and implementation layers by applications
 - pz – these are hidden properties used internally by the system that we cannot see or update
 - Describe the property modes that can be selected in a property rule
 - Three types
 - Value
 - Value list
 - Value group
 - If repeated in the work type:
 - Page
 - Page List
 - Page Group
 - Single value property types
 - Date
 - DateTimeDecimal
 - Double
 - Integer

- Text
 - Identifier
 - Password
 - True or False
 - Time of Day
- Understand the difference between the single, list and group types for the value and page modes
 - Page values referenced by .Page.Property
 - Page list values reference by .Page(n).Property
 - Page group values referenced by .Page(string).Property
- Define an embedded page
 - This is created by associating a work type property with a data class, setting the property mode to page
- Name the syntax used to reference pages and properties
 - See above
- Define a property that prompts a user to select a value
 - PromptSelect
 - This can use a local list or a data table to populate the list
 - Set property mode to single value
 - Set HTML property to PromptSelect
 - Table type to local list or the data table
 - DynamicSelect
- Use the properties wizard to define multiple properties
 - Pops up a wizard containing a table through which multiple properties can be defined
 - First page enters Description, Name, Mode, Type
 - Second page enters Display as, Details related to display as
- Use the data table wizard to create a table and its associated instances
 - Used for reference data that may change from time to time
 - Created/Updated through the data table wizard
 - Always has an ID column – all rows have a unique ID
 - Defined in a concrete data class to allow it to be persisted
 - Used for PromptSelects and dynamic list controls
- Models
 - Define models to initialize data
 - These are in the tech category
 - Under model there is pyDefault
 - The properties on a work item that need to be initialised to a specific value are defined in pyDefault
 - Describe how superclass model chaining works
 - pyDefault is inspected for the work type – any properties initialised in the worktype are initialised using rules in this class
 - The normal method is used for exploring the class hierarchy, if any more properties are initialised in the superclasses they will be initialised at the most specialised level at which the initialisation is defined.

Automating Business Policies

■ Decision rule types provided by PRPC

- Decision Rule Types
 - Describe the following types of decision rules and their function
 - Decision tables
 - A decision table has the properties to be tested across the columns
 - Each when row has conditions against one or more of the properties and an action (or return value) for that row
 - If the row is evaluated to be true then the action is completed
 - Has an otherwise row at the end for all other outcomes – this has an action
 - Can set allowable results which constrain the allowed actions/return values
 - Should be used when evaluating against a small set of properties and when most of these properties are used several times, the logic is easy to follow and business people like them
 - Decision Trees
 - Like a more standard if statement
 - If condition then continue/return – continue to go into a nested if clause, return to return a value out of the decision tree
 - Use otherwise at the end of each clause (like an else)
 - The actions that occur when a clause is true can return a result or call an action function (for example setPropertyValue)
 - When conditions
 - These evaluate a one or more logical tests and return true or false only
 - Can be called by other rules or shapes – flow actions, activities to evaluate pre-conditions and transitions, UI rules for conditional display
 - Can combine clauses using and or or
 - Can use expression builder to create complex clauses
 - Can build complex, nested clauses using the advanced options and the Logic of Table Elements field – nest using brackets
 - All are listed in the decision category

■ Using decision rules in applications

- Fork and decision shapes
 - Explain the difference between fork shapes and decision shapes in a flow
 - Fork shape
 - This is used for when rules
 - Each connector out of the fork has a when rule associated with it
 - Each connector has a percentage assigned – these are used to determine the order in which the when clauses are evaluated
 - Always use an else connector – this is used when all when clauses are false
 - Decision shape
 - This is used for decision tables and trees
 - The connectors are generated from the decision tree or table

■ Declarative processing

General Functionality

- Understand the difference between procedural and declarative programming models
 - Procedural follows steps – standard programming model, all steps required have to be explicitly called by one of the other steps
 - Declarative does not, instead it is automatically invoked when conditions are fulfilled
- Describe the benefits of declarative processing
 - Are automatically run when a value is set or changes
 - Do not need to be called explicitly
 - Do not need to be run in a sequential fashion
 - Allow relationships between properties to be expressed directly without the need to create and call activities
 - Model the relationship visually and let the system do the programming
 - Speed development and simplify maintenance (much easier to find the procedural code as they are directly associated with the property)
- Use the Dependency Analysis Tool (DNA)
 - Set rules inspector to label declarative rules
 - Click on the D
 - DNA is launched
 - Shows the declarative expression as a decision tree – values can be updated for testing purposes, the decision tree will evaluate
- Rule types
 - Identify the various computations for Declare Expressions
 - Value of
 - Sum of
 - Minimum of
 - Maximum of
 - Average of
 - Count of
 - Result of Decision Tree
 - Result of Decision Table
 - Result of Map Value
 - Describe how the declare expressions can be dynamically called
 - These are called when a property that references a declarative rule changes
 - These are performed on the server
 - They are enabled through selecting Enable Expression Calculation on the HTML tab of a harness or row section
 - Types of Declaratives
 - Constraints (Rule-Declare-Constraints)
 - Define relationships between property values that have to be true at all times – prevents a user from progressing if this is false
 - Declare Expression (Rule-Declare-Expression)
 - Automatically performs a calculation in response to one of the values in the expression changing
 - Declare Pages

- Creates read only clipboard pages on demand and then refreshes as specified – these must start with Declare_
- Declare Index
 - Extracts a value from an embedded property when its value has changed, allows it to be reported upon and improves performance
- Declare OnChange
 - Is run if the specified property is changed
- Declare Trigger
 - Is run when instances of a specific class are created, updated or deleted in the DB

■ Service Levels Purpose

Service level rules define three time intervals, known as goal, deadline, and late intervals. The goal time is the smallest time interval, the deadline time is a longer interval, and the late interval defines post-deadline times. Each time interval is in days, hours, minutes, and seconds.

Service level rules can be associated with a work item or an assignment. When the time interval defined by the service level is reached without the assignment being performed (or the work item becoming resolved), escalation occurs.

Escalation can change the urgency value of the assignment or work item, send it to someone else, send an alert or email message, cancel the work, or initiate other processing.

Where referenced

Service level rules are referenced in the assignment shapes of flows.

The Pega-ProCom agents rule — a background requestor — detects goals and deadlines not met and performs escalation processing. To make your application's custom service level rules visible to this agent, update the access group for the Pega-ProCom agent schedule data instance to include your application RuleSet versions.

Service level rules are part of the Process category. A service level rule is an instance of the **Rule-Obj-ServiceLevel** rule type. Service level rules are sometimes called **service level agreements**, or SLAs.

Creating Messages and Notifications

■ Correspondence rule types

- Correspondence
 - Identify the different correspondence types supported by PRPC
 - Email
 - Letter
 - Phone text
 - Fax
 - List common activities used to generate correspondence
 - CorrNew
 - CorrQuickStart
 - Input parameters
 - CorrName – name of the correspondence rule (defines the content of the correspondence, can use properties in the correspondence to personalise)

- PartyRole – the role of the person to whom this will be sent
- Broadcast – optional, send to all parties associated with the work object

■ Sending correspondence from flows

- Can use a notify shape to send correspondence
 - Used to send automatic notification messages, use with standard notify activities
 - Specify partyrole, rule used to send correspondence and corrName in the properties panel
- Can use a utility shape to send correspondence
 - Can be used for any kind of correspondence
 - Use with standard CorrNew activity
 - Set rule in properties to CorrNew, then enter parameters
- Can send from assignments?
 - This will ask the user to enter the partyrole/URI for the recipient

■ Work parties

- Explain the purpose of a work party
 - Work party is a page group on the work item
 - There are standard out of the box work party classes:
 - Data-Party-Gov
 - Data-Party-Operator
 - Data-Party-Com
 - Data-Party-Org
 - Data-Party-Person
 - can be extended – i.e. Data-Party-XXX
 - Embedded in pyWorkPage as a page group, the identifier for the group is the Role name set in the work parties form – e.g. pyWorkParty(Customer)
 - The work parties can be of the OTB types but other types can be created
 - Work parties are the entities that are interested in the progress of a work item – a work item may have any number of associated work parties
 - The work party may not have any responsibility for progressing the work item
 - Roles determine a work party's involvement in the process – interested, originator for example
 - Party roles are defined in a work parties rule – named default by convention and are in the Process category
 - Default Work Parties are created by AA or by completing the work parties form
 - In work parties form enter label, role, party class, model (for initialisation)
 - Can route to a work party using rule ToWorkParty on a router
 - Can add parties as a local action in an assignment

Building Reports

■ Using the Report Browser

■ Using the Report Editor to create or modify reports

- Report types and capabilities

<https://pdn.pegacom/reporting/comparing-report-definition-list-view-and-summary-view-reports>

- List-type reports

- Summarized reports

- Charting

- Using PRPC data in reports

<https://pdn.pegacom/reporting/when-to-use-and-when-not-to-use-unoptimized-properties-in-reports>

Business Architect Best Practices

- Ten Guardrails for successful application development

1. Adopt an Iterative Approach

Define an initial project scope that can be delivered and provide business benefit within 60-90 days from design to implementation.

Document five concrete use case scenarios up front and evaluate them at the end to calibrate benefits.

Use your scenarios as story boards and ensure that each delivers a measurable business benefit.

2. Establish a Robust Foundation

Design your class structure complying with the recommended class pattern.

It should be understandable, be easy to extend, and utilize the standard work and data classes appropriately.

Use your organization entities as a starting pattern, and then proceed with class groups.

Lead with work objects. Create the class structure and “completed work” objects early.

Position rules correctly by class and/or RuleSet.

Actively use inheritance to prevent rule redundancy.

3. Do Nothing That is Hard

Use “out of the box” functionality as much as possible, especially in the initial project release.

Avoid creating custom HTML screens or adding buttons.

Always use the “Auto Generated HTML” feature for harness sections and flow actions.

Always use the standard rules, objects, and properties. Reporting, Urgency, Work Status, and other built-in behaviors rely on standard properties.

Never add a property to control typical work or for managing the status or timing of work.

4. Limit Custom Java

Avoid Java steps in activities when standard Process Commander rule types, library functions, or activity methods are available.

Reserve your valuable time and Java skills for implementing things that do not already exist.

The top screenshot shows an activity editor for 'ACTIVITY InsCorp-PL-HomePA-Work-Policy • ClaimHistory'. It has tabs for 'Steps', 'Parameters', 'Pages & Classes', 'Security', and 'History'. A table lists steps with columns for 'Label', 'Description', 'Step', and 'Method'. Step 1 is 'Invoke the connector' with method 'Java'. A red circle with a white slash is overlaid on the 'Step' field.

The bottom screenshot shows the same activity editor with a different set of steps. A green circle with a white checkmark is overlaid on the 'Method' column. The steps are:

Label	Description	Step Page	Method
1.	Create the data page	DataPage	Page-New
2.	Map the request data	DataPage	Property-Set
3.	Invoke the connector	DataPage	Connect-SOAP
4.	Map response parameters	DataPage.getClaims_getClaims	Property-Set
5.	Delete the data page	DataPage	Page-Remove

5. Build for Change™

Identify and define 10-100 specific rules that business users own and will maintain.

Activities should not be on this list. Use other rule types for business-maintained logic.

The screenshot shows a 'DECISION TABLE InsCorp-PL-HomePA-Work-Policy • BasePremium'. It has tabs for 'Table', 'Results', 'Pages & Classes', and 'History'. Buttons for 'Show Conflicts', 'Show Completeness', and 'Edit in Excel' are visible. The table below is divided into 'Conditions' and 'Actions'.

	YearBuilt		Appraised Market Value	Area	Return
	>=	<=			
if	1983	2008	<500000	<2500	⇒ 300
else if	1983	2008	>=500000		⇒ 400
else if	1958	1982	<500000	<2500	⇒ 450
else if	1958	1982	<500000		⇒ 600
else if	1958	1982	>=500000		⇒ 750
otherwise					⇒ 500

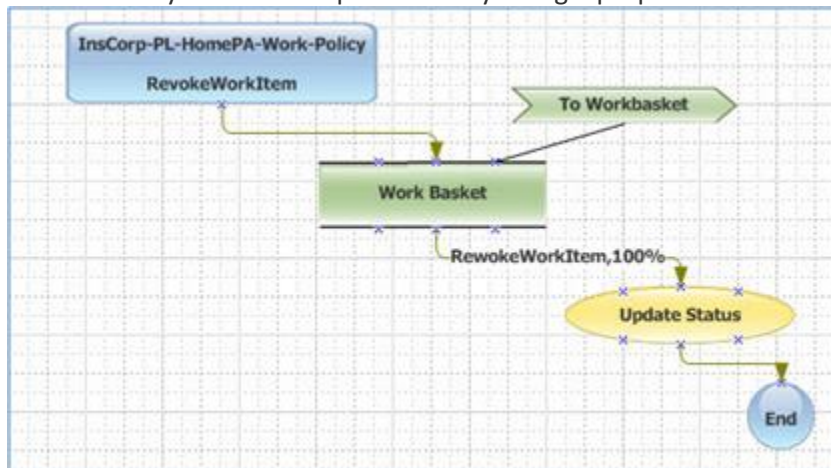
6. Design Intent-driven Processes

Your application control structure must consist of flows and declarative rules, calling activities only as needed.

Use flow actions to prompt a user for input.

Present fewer than five connector flow actions for any individual assignment. If you need more than that, you need to redesign the process.

Create activity rules that implement only a single purpose to maximize reuse.



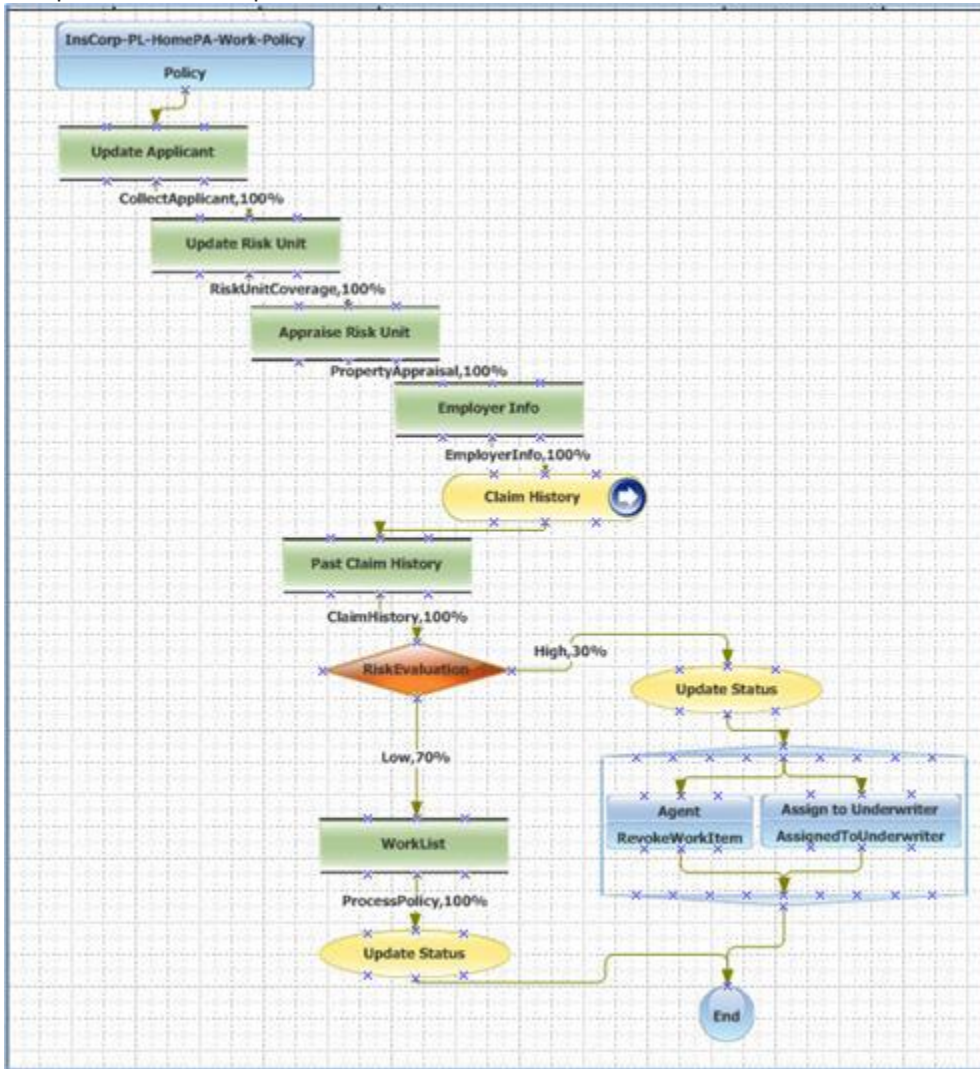
7. Create Easy-to-Read Flows

Your flows must fit on one page and must not contain more than 15 SmartShapes (excluding Routers, Notify shapes and Connectors) per page.

If a flow has more than 15 SmartShapes:

Create a subflow.

Use parallel flows to perform additional functions



8. Monitor Performance Regularly

Evaluate and tune application performance at least weekly using Performance Analyzer (PAL) to check rule and activity efficiency.

Use PAL early to establish benchmarks. Compare these readings to follow on readings; correct application as required.

9. Calculate and Edit Declaratively, Not Procedurally

Whenever the value of a property is calculated or validated, you must use declarative rules wherever appropriate.

Create a Declare Expressions rule instead of using a Property-Set method in an activity.

Use a Declare Constraints rule instead of a Validation rule.

10. Keep Security Object-Oriented, Too

Your security design must be rule-based and role-driven based on who should have access to each type of work.

Never code security controls in an activity.

Use the standard access roles that ship with Process Commander only as a starting point.

Use RuleSets to segment related work for the purpose of introducing rule changes to the business, not as a security measure.

■ Delegating rules to the business and Build for Change®

A **delegated rule** allows you to create a quick shortcut to rules or data instances. The links to delegated rules are referred to as [favorites](#). Click the link to open the form. Access depends on which portal you are using.

These links provide quick access as follows:

- From the Designer Studio, these links are displayed in the Private Explorer. You can also see your delegated rules by selecting **My Favorites** from your Profile menu.
- For users who use the *pyCaseManager* portal, selecting **Options > My Rules** displays these links on the **My Business Rules** tab.
- For managers who use the composite *Manager* portal, these links are displayed in the **Delegated Rules** gadget.
- For managers who use the traditional ('fixed') *WorkManager* portal, these links are displayed on the **My Business Rules** area in the Dashboard workspace. Depending on access roles and privileges, the manager can update the entire rule form or only fields in the leftmost tab of the rule form.

A rule can be delegated to a single user or to an access group.


Delegating a rule (making a favorite)

To mark a rule as a favorite (delegated), open the rule form and select [Actions > Add to Favorites](#) menu on the toolbar.

Complete the dialog box:


1. In the **Label** field, enter a text label for this rule. This label appears in the Private Explorer and in [My Favorites](#) on your Profile menu. The **Short Description** of the rule is used as the default. Keep the **Label** value short. (You are not required to choose a **unique** label; two favorite rules can have the same label.)
2. In the **Add To** field, select [My Personal](#) or [My Access Group](#).
3. Select **Open the Highest Version** if you want the system to find the rule using the rule resolution algorithm. Select **Always Open This Version** box selected if you want this exact version, in this exact RuleSet, to be delegated. This option does not appear on rules that are not versioned such as application or class.
4. Click **OK**.

A best practice


 To segregate delegated rules — those **expected** to change from time to time, especially by managers using the application — from the larger collection of undelegated rules in your application (which usually belong to locked RuleSet versions), dedicate a RuleSet and Version to them. Copy the rules to be delegated into this RuleSet version, which can remain unlocked. As a convenience, you can choose to not require check-out for this RuleSet.

Editing or removing a favorite or delegated rule

To edit or remove your own favorites (set for yourself or your access group):

- In your Profile menu, select [My Favorites > Edit Favorites](#). In the Favorites window, edit the label or click  to delete the favorite. Click **Submit**.
- In the Private Explorer, select the favorite, right click, and select [Delete this favorite](#).
- In the [pyCaseManager](#) portal, click the **Edit** button in the **My Business Rules** tab. Complete the dialog box and click **Submit**.
- In the [WorkManager](#) portal, click the **Edit** button in the **My Business Rules** area of the Dashboard workspace. Complete the dialog box and click **Submit**.
- In the [Manager](#) portal, click the **Edit** button in the **Delegated Rules** area of the Dashboard workspace. Complete the dialog box and click **Submit**.

To remove the delegation from **another** user or from another access group:

1. In the Application Explorer, enter `System-User-MyRules` in the autocomplete field to list all instances of the [System-User-MyRules](#) class.
2. Locate the instance of the user or access group and select it to open the Favorites rule form.
3. Select the favorite you want to remove and click  to delete it from the form.
4. Save the form.

Notes

- Delegation of a rule doesn't eliminate the need for the delegate to hold appropriate access roles and privileges to check out the rule (if required) and check it back in.
- You cannot delegate a checked out rule.
- One rule may be simultaneously delegated to several users or access groups.
- Any Designer Studio user who has an appropriate access role can open a rule, whether delegated or not, through the Explorer tools.
- To remove a delegation, update the Favorites form (*System-User-MyRules* class instance) for that operator or access group.
- You can delegate a specific circumstance-qualified rule or time-qualified rule. For example, if your application includes a base rule and 50 circumstance-qualified rules that qualify it — one for each state in the United States — you can delegate the Vermont rule to Smith, the California rule to Schwarzenegger, and so on.

How to build for change

Benefits

Business environments change, sometimes suddenly, in response to external change or internal decisions. PRPC allows you to design and implement applications that respond to change with agility and efficiency.

By recording business practices and policies in rules rather than in code, all PRPC applications provide a degree of modularity and transparency that can simplify maintenance. For example, line managers can review and comment on a business process presented as a Process Modeler diagram, even without knowing Process Modeler how to maintain a flow rule.

To go beyond this basic PRPC benefit, application developers can delegate responsibility for updating selected important parts of each application to business managers. This promotes:

- Agile response to changing business conditions
- Reduced workload for application developers of minor, low risk maintenance items
- Shared responsibility for the success of production applications
- A degree of empowerment of those closest to day-to-day operations
- Experimentation and process innovation by line managers

Which rules can managers update?

who have the *PegaRULES:WorkMgr4* access role (or other roles that include the *UpdateLimitedForm* privilege) can view and update the leftmost tab of any delegated rule. They can also view and update the **History** tab.

By design, the leftmost tab of the forms for ten rule types provides access to fields most likely to change over time. Although rules of any type can be delegated, managers are most likely to learn and understand the following types:

- Case match
- Constraints
- Decision table
- Decision tree
- Declare Expression
- Flow
- Flow action
- Service level
- Map value
- When condition

Designing for delegation

To empower managers to change selected rules:

1. Identify during design which rules are useful to delegate to managers to maintain.
2. Choose rule names and property names that are meaningful in the business context.
3. Choose **Short Description** text carefully for properties that may be referenced in the rules.
4. After initial testing is complete, copy the rules into a RuleSet containing only the delegated rules. This segregates any later changes to them from the rest of the application.
5. Delegate the rules, and confirm that they can be accessed from the manager's dashboard, with meaningful labels.
6. Provide appropriate training and documentation to line managers.