الجمهورية العربية السورية
المعهد العالي للعلوم التطبيقية والتكنولوجيا

ماجستير نظم المعلوماتية ودعم القرار

مضر رفعت كيوان

أُعدت هذه الأطروحة لنيل

شهادة الماجستير في اختصاص نظم المعلوماتية ودعم القرار

من نموذج إجرائية العمل إلى خدمات الويب

بإشراف

د.غيداء ربداوي **د.كادان الجمعة**

دمشق في 22 أذار 2016

إهداء

إلى من أقتسم معها كل بسمّةٍ أو دمعة...

إلى زوجتي الغالية غيثاء

إلى أحلى مفاجأةٍ خبّأها ليَّ القدر...وأغلى هديّة حملّتها ليَّ الأيام....

إلى ملاكي الصغيرة صوفيا

إلى تلك الدنيا التي أحتضنتني لتغمرني بدفئها، ولتغدق عليَّ بالحب والحنان....

والدتي

وإلى صديق الحياة إلى الأمل الذي لا يذوي أبداً....

والدي

شكر

شكر لكل من ساهم في تحقيق هذه الأطروحة وساعد في إنجاحها.....

كما أتوجّه بجزيل الشكر للدكتورة غيداء ربداوي - باحثة في المعهد العالي للعلوم التطبيقية والتكنولوجيا – لكل ما قدمته من توجيهات ونصائح وأفكار ودعم لإتمام هذا العمل.

و أتوجّه بخالص الشكر أيضا للدكتور كادان الجمعة - باحث في المعهد العالي للعلوم التطبيقية والتكنولوجيا - لكل ما قدمه من أفكار ونصائح وإشراف ودعم لإتمام هذا العمل.

ملخص

يشكل التمثيل الرمزي لإجرائيات العمل النموذج الأساسي المستخدم من قبل محللي الأعمال لتوصيف إجرائيات العمل ضمن الشركات والمنظمات، هذه الإجرائيات التي ترغب معظم الشركات بتحويلها إلى خدمات ويب تكون متاحة للأفراد والشركات الأخرى المتعاونة معها. إلا أنَّ التمثيل الرمزي غير مضبوط بشكل جيد لاستخدامه من قبل التقنيين ومطوري خدمات الويب والذين يستخدمون اللغة التنفيذية لإجرائيات العمل. ونتيجة لاختلاف طبيعة التمثيل الرمزي واللغة التنفيذية ومجال استخدامهما فقد نشأت فجوة كبيرة بينهما، وجرت عدة أبحاث ووضعت عدة استراتيجيات للانتقال بين التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل، ويأتي هذا البحث ضمن تلك الأبحاث. تستعرض هذه الأطروحة منهجيات التحويل السابقة وتقارن بينها وتناقش نقاط ضعف وقوة كل منها، كما تطرح نموذجاً مطوراً للحل مبني على خوارزمية إيشويس-كريفن[1] التي تقوم بتجميع الخدمات ضمن إجرائيات مهيكلة، فيتم تعديل هذه الخوارزمية ليكون دخلها التمثيل الرمزي لإجرائيات العمل وخرجها اللغة التنفيذية لإجرائيات العمل، كما تمت إضافة خوارزمية لمعالجة مشكلة الروابط المتزامنة بين الفروع المتوازية التي كانت تعاني منها خوارزمية إيشويس.

تشترط الخوارزمية التي انطلق منها هذا البحث كما معظم خوارزميات التحويل السابقة أن يكون دخلها مخطط إجرائية عمل مبني بشكل جيد، لذلك تمّت دراسة شروط البناء الجيد لمخطط إجرائية العمل، ومناقشة إمكانية التراخي ببعض هذه الشروط وأثر ذلك على نموذج الحل. وفي نهاية الأطروحة تم ذكر بعض الأفكار لدراستها وتنفيذها مستقبلا بهدف تطوير عملية التحويل بين التمثيل الرمزي واللغة التنفيذية.

كلمات مفتاحية

مخطط إجرائية العمل، خدمات ويب، تمثيل رمزي لإجرائيات العمل، اللغة التنفيذية لإجرائيات العمل، تجميع الخدمات، الإجرائيات المهيكلة، الروابط المتزامنة، مخطط إجرائية العمل المبني بشكل جيد

[1] خوارزمية إيشويس-كريفن أو خوارزمية تجميع الخدمات ضمن إجرائيات مهيكلة [16].

Abstract

Most business analyzers use the Business Process Modeling Notation (BPMN) to describe business processes, which almost all companies need to transform to available web services.

Technicians who develop web services cannot use BPMN because it is not well structured to describe Web services. They use many more structured languages as Business Process Execution Language (BPEL).

The two languages (BPMN, BPEL) have different nature and they are used in two different scope, so there is a large gap between them. There are several researches and strategies to map between BPMN, BPEL. Our research comes within these researches.

In our thesis, we present many previous approaches for mapping BPMN to BPEL. Moreover, we compare between their own strengths and weaknesses. Then we offer our new approach based on ESHUIS-GREFEN[2] algorithm. We develop this algorithm to accept BPMN as input and BPEL as output. In addition, we solve the synchronization problem in ESHUIS algorithm by adding algorithm to find and process cross-synchronization links between parallel branches. We evaluated our approach by doing two case studies and comparing the results with other approaches and implementations.

Our approach and most of previous approaches use well-formed business process diagram as an input, so we evaluated the well-formed restrictions and conclude that some of these restrictions can be relaxed. At the end, we presented some ideas for future work.

Keywords

Business process diagram, web services, BPMN, BPEL, composing services, structured process, synchronization dependencies, well-formed business process diagram

[2] R. Eshuis and P.W.P.J. Grefen. Composing services in to structured processes [16].

قاموس المصطلحات

الاختصار	المفهوم الإنكليزي	المفهوم العربي
BPEL	Business Process Execution Language	اللغة التنفيذية لإجرائيات العمل
BPMN	Business Process Modeling Notation	التمثيل الرمزي لإجرائيات العمل
BPD	Business Process Diagram	مخطط إجرائية العمل
OMG	Object Management Group	منظمة الإدارة الغرضية
OASIS	Advancing Open Standards for the Information Society	منظمة النهوض بمعايير المعلومات المنظمة
SOA	Service Oriented Architecture	البنية الموجهة بالخدمات
UDDI	Universal Description, Discovery and Integration	الوصف والاكتشاف والتكامل العام لخدمات الويب
WSDL	Web Service Description Language	لغة توصيف خدمات الويب
EPCs	Event-driven Process Chain	سلاسل الإجرائية المقادة بالحدث
SESE-regions	Single-Entry Single-Exit regions	مناطق ذات نقطة دخل واحدة ونقطة خرج واحدة
DPE	Dead-Path-Elimination	حذف مسار الموت
DOM	Dominators	العقدة المهيمنة
HEAD	loop headers	رأس الحلقة
FOLLOW	follow-sets	المجموعات اللاحقة

قائمة الأشكال

الشكل (2.1) العناصر الأساسية للتمثيل الرمزي لإجرائيات العمل 20
الشكل (2.2) عناصر مخطط إجرائية العمل وأنواعها 21
الشكل (2.3) مخطط الصفوف لعناصر التمثيل الرمزي وعلاقتها ببعضها 22
الشكل (2.4) مخطط إجرائية عمل لمعالجة شكوى 26
الشكل (2.5) تمثيل منهجية الحفاظ على العنصر.....33
الشكل (2.6) تمثيل لخوارزمية تصغير العنصر.....34
الشكل (2.7) تمثيل لخوارزمية تطابق البنية 34
الشكل (2.8) تمثيل لخوارزمية توسيع البنية 35
الشكل (2.9) تمثيل لخوارزمية قواعد حدث-شرط-فعل 36
الشكل (2.10) تحويل النماذج المبنية بشكل جيد إلى رماز اللغة التنفيذية 37
الشكل (2.11) تحويل النماذج شبه المهيكلة إلى نماذج مهيكلة ضمن منهجية أويان 38
الشكل (2.12) تحويل مخطط إجرائية إلى بيان ضمن خوارزمية تشي-زينغ 40
الشكل (2.13) نماذج صنعية توصف عناصر BPMN الأساسية ضمن البيان E-EGGs.....40
الشكل (2.14) بيان وسيط للانتقال إلى اللغة التنفيذية ضمن خوارزمية تشي-زينغ 41
الشكل (2.15) مراحل دمج عقد البيان للوصول إلى رماز اللغة التنفيذية ضمن خوارزمية تشي-زينغ 41
الشكل (2.16) تحويل مناطق (SESE) إلى شجرة الإجرائية المهيكلة 42
الشكل (2.17) بيان التبعية لإجرائية طلب أسعار ودفع فاتورة 44
الشكل (2.18) التركيب المهيكل الناتج عن تطبيق خوارزمية التجميع 44
الشكل (3.1) نموذج الحل المقترح لعملية التحويل بين التمثيل الرمزي واللغة التنفيذية 57
الشكل (3.2) روابط متزامنة خاطئة تؤدي إلى قفل الموت 58
الشكل (3.3) خوارزمية إيجاد الروابط المتزامنة 62
الشكل (3.4) مخطط إجرائية معالجة الشكوى 63
الشكل (3.5) خوارزمية بناء المكونات المهيكلة 68
الشكل (3.6) خوارزمية إضافة الروابط المتزامنة 69
الشكل (3.7) خوارزمية إدخال عقدة تابعة وحيدة بعد العقدة الحالية 70
الشكل (3.8) التراكيب المهيكلة الناتجة لمخطط إجرائية معالجة الشكوى 71

الشكل (3.9) نموذج BPMN مرسوم باستخدام الإضافة ضمن Eclipse 74
الشكل (3.10) رسم توضيحي لعمل التنجيز البرمجي 75
الشكل (4.1) الحالة المدروسة الأولى – طلب قرض 80
الشكل (4.2) التركيب المهيكل للحالة المدروسة الأولى 83
الشكل (4.3) بيان التبعيات الممثل للحالة المدروسة الأولى 84
الشكل (4.4) شجرة الإجرائية المهيكلة للحالة المدروسة الأولى 85
الشكل (4.5) جزء من رماز اللغة التنفيذية للحالة المدروسة الأولى 86
الشكل (4.6) الحالة المدروسة الثانية: حساب المخاطر الكلية لبطاقة الائتمان 88
الشكل (4.7) التركيب المهيكل للحالة المدروسة الثانية 91
الشكل (4.8) بيان التبعيات الممثل للحالة المدروسة الثانية 93
الشكل (4.9) الإجرائية المهيكلة للحالة المدروسة الثانية..... 94
الشكل (4.10) جزء من رماز اللغة التنفيذية للحالة المدروسة الثانية 95
الشكل (5.1) الحالة المدروسة الأولى بعد تحويلها باستخدام الإضافة (BPMN2BPEL) 100
الشكل (5.2) نموذج الحالة المدروسة الأولى بعد تعديله ليناسب الإضافة (BPMN2BPEL) 100
الشكل (5.3) الحالة المدروسة الأولى بعد تحويلها باستخدام الإضافة (BABEL) 101
الشكل (5.4) مخطط إجرائية عمل الحالة المدروسة الأولى بعد تعديله ليناسب الإضافة (BABEL) 101
الشكل (5.5) الحالة المدروسة الثانية بعد تحويلها باستخدام الإضافة (BPMN2BPEL) 102
الشكل (5.6) نموذج الحالة المدروسة الثانية بعد تعديله ليناسب الإضافة (BPMN2BPEL) 102

قائمة الجداول

الجدول (2.1) توضيح لعناصر اللغة التنفيذية لإجرائيات العمل 28
الجدول (2.2) المقارنة بين التمثيل الرمزي واللغة التنفيذية من حيث دعم نماذج التدفق 31
الجدول (2.3) الاستراتيجيات الحالية المستخدمة لعملية التحويل بين التمثيل الرمزي واللغة التنفيذية 49
الجدول (2.4) مختصر المقارنة بين الاستراتيجيات الحالية المستخدمة لعملية التحويل 51
الجدول (3.1) المجموعتين before(g)-set, after(g)-set لبوابات مخطط إجرائية العمل 63
الجدول (3.2) العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة 71
الجدول (3.3) تحويل التركيب المهيكل إلى رماز اللغة التنفيذية 72
الجدول (4.1) المجموعتين before(g)-set,after(g)-set للبوابات لحالة الاستخدام الأولى 81
الجدول (4.2) العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة لحالة الاستخدام الأولى 82
الجدول (4.3) المجموعتين before(g)-set,after(g)-set للبوابات لحالة الاستخدام الأولى 89
الجدول (4.4) العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة لحالة الاستخدام الثانية 90
الجدول (5.1) المقارنة بين النموذج المقترح والأداتين (BABEL, BPMN2BPEL) 103

الفهرس

الفصل الأول 12

1. المقدمة 13
1.1. إشكالية البحث 14
1.2. أسئلة البحث 15
1.3. مقترح الحل 16
1.4. فرضية البحث 16
1.5. مخطط البحث 16

الفصل الثاني 17

2. دراسة مرجعية 18
2.1. التمثيل الرمزي لنمذجة إجرائية العمل 18
2.2. اللغة التنفيذية لإجرائية العمل 27
2.3. الأبحاث السابقة للتحويل بين التمثيل الرمزي واللغة التنفيذية 30
2.3.1. الاختلاف بين التمثيل الرمزي واللغة التنفيذية لإجرائية العمل..... 30
2.3.2. أخر الأبحاث في مجال التحويل بين التمثيل الرمزي واللغة التنفيذية 32
2.3.3. دراسة نقدية 45
2.3.4. نقاط القوة والضعف لمنهجيات التحويل السابقة 46
2.3.5. الأدوات التنفيذية للتحويل المتوفرة حالياً 53
2.4. الخاتمة 54

الفصل الثالث 55

3. نموذج الحل المقترح 56
3.1. مقدمة 56
3.2. نموذج الحل 56
3.2.1. تبعيات التزامن 58
3.2.2. العقدة المهيمنة، رأس الحلقة، المجموعات اللاحقة 64
3.2.3. قواعد اللغة التنفيذية BPEL 65
3.2.4. الخوارزمية 66
3.2.5. نحو اللغة التنفيذية لإجرائية العمل 72
3.3. التنجيز البرمجي لنموذج الحل 73
3.3.1. بناء نماذج التمثيل الرمزي لإجرائيات العمل باستخدام الEclipse 73
3.3.2. تنجيز الخوارزمية 74

الفصل الرابع 78

4. الحالات المدروسة 79
4.1. الحالة المدروسة الأولى: معالجة طلب قرض 79
4.1.1. التحويل اليدوي 79
4.1.2. التحويل باستخدام التنجيز البرمجي 83
4.2. الحالة المدروسة الثانية: حساب المخاطر الكلية لبطاقة الائتمان 87
4.2.1. التحويل اليدوي 89
4.2.2. التحويل باستخدام التنجيز البرمجي 92

الفصل الخامس 96

5. تقييم نموذج الحل 97
5.1. نقاط القوة والضعف 97
5.2. المقارنة مع باقي المنهجيات والبرمجيات 98
5.2.1. الحالة المدروسة الأولى 99
5.2.2. الحالة المدروسة الثانية 101
5.2.3. الخاتمة 103
5.3. مناقشة شروط البناء الجيد لمخطط إجرائية العمل 104

الفصل السادس 108

6. الخاتمة والآفاق المستقبلية 109
6.1. الخاتمة 109
6.2. الآفاق المستقبلية 110

المراجع 111

الفصل الأول

المقدمة

1. المقدمة

لقد تطورت شبكة الانترنيت لتشمل مصادر المعلومات المختلفة ولتضعها بمتناول جميع المستخدمين وفي كافة مجالات الحياة، هذا التطور دفع مختلف المؤسسات والمنظمات في العالم لتقديم خدماتها عن طريق الشبكة العنكبوتية والذي أدى بالضرورة إلى النمو السريع والمتعاظم لهذ الشبكة، ونظراً لوجود التعاون والتفاعل بين مختلف المؤسسات والمنظمات فقد ظهرت الحاجة لوجود منصة إلكترونية لهذا التعاون والتفاعل، فظهرت خدمات الويب لتأمين هذا التفاعل والتخاطب ولتسمح للشركات بتقديم خدماتها بطريقة سلسة ومنظمة وسهلة لتستطيع الجهات الأخرى الاستفادة منها دون الحاجة إلى فهم عميق بعمل الشركة أو إجرائياتها أو حتى طريقة بناء الخدمة. فتستطيع الجهات الأخرى سواء أكانت منظمة أو أشخاص أو حتى برنامج إلكتروني الاتصال بخدمة الويب والتخاطب معها والحصول على النتيجة التي تريدها عن طريق بروتكولات التخاطب والاتصال الخاصة بهذه الخدمة. وفي ضوء الخدمات الكثيرة والمتنوعة والمنتشرة بشكل واسع كان لابد من البحث عن طرق أكثر فعالية للوصول إلى الخدمات المطلوبة وبأسهل الطرق. فنشأت اللغة التنفيذية لإجرائيات العمل (BPEL)، حيث يوجد العديد من المنصات والبيئات التي تدعم تنفيذ إجرائيات اللغة التنفيذية وبعض هذه المنصات تدعم التعريف البياني لهذه الإجرائيات مثل (Oracle BPEL Process Manager, IBM Web Sphere Application)، وبالرغم من ذلك فإن هذه الأدوات لا تناسب محللي الأعمال الذين سيقومون ببناء وتصميم إجرائيات عمل الشركة التي ستتحول إلى خدمات ويب. ذلك أن هذه المنصات تعتمد النحو الخاص باللغة التنفيذية غير المناسب لمحللي الأعمال والذين يعتمدون على لغات بيانية أكثر قرباً لمجال عملهم ومخصصة لتوصيف إجرائيات العمل كالتمثيل الرمزي لإجرائيات العمل (BPMN) والتي تشكل لغة وسيطة بين محللي الأعمال ومهندسي النظم لتعريف إجرائيات العمل ليتم تنفيذها لاحقاً. وبالرغم من وجود الكثير من الأدوات التي تدعم النمذجة البيانية لإجرائيات العمل، إلا أن هذه الأدوات لا تدعم تنفيذها مباشرة، لذلك ظهرت الحاجة لوجود طريقة للانتقال من النمذجة البيانية إلى اللغة التنفيذية لإجرائيات العمل. وقد تم بناء العديد من النماذج والأدوات للانتقال بين اللغتين، وجرت المقارنة بينها على أساس قدرتها على تحويل أي نموذج وكذلك مقدار وضوح وقابلية قراءة الرماز الناتج عن عملية التحويل بسبب الحاجة إلى تتبعه ومقارنته يدوياً في بعض الحالات.

إن ضعف عملية التحويل بين النمذجة البيانية واللغة التنفيذية لإجرائيات العمل هو أمر متوقع بالنظر إلى الطبيعة المختلفة للغتين باعتبار الأولى لغة توصيفية تعتمد على الرسوميات بتوصيفها، والثانية لغة تنفيذية تعتمد على بنية كتلية في تركيبتها. يقدم هذا العمل عملية التحويل بين التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل مع الأخذ بعين الاعتبار المتطلبات السابقة مع مراعاة أن يكون رماز اللغة التنفيذية الناتج عن التحويل قابلية القراءة ومفهوم من قبل البشر.

1.1. إشكالية البحث

عمل الكثير من الباحثين سابقاً على تطوير خوارزمية لإجراء التحويل من التمثيل الرمزي لنمذجة إجرائيات العمل إلى اللغة التنفيذية لإجرائيات العمل [5]، [6]، [7]، [8]، [9]، [10]، [15]، وقد ظهرت الكثير من المشاكل في هذا التحويل بسبب اختلاف طبيعة التمثيلين ومجال عمل كل منهما، ومع المحاولات الكثيرة تم وضع عدة طرق ومنهجيات للانتقال السابق. ولكن رغم تعدد هذه الأدوات والمنهجيات إلا إنها عجزت عن تحقيق العديد من المتطلبات:

i. **الكمال (completeness)**: إمكانية تحويل جميع نماذج إجرائيات العمل إلى لغة تنفيذية.
ii. **الأتمتة (automation)**: انتاج الرماز المطلوب بدون تدخل بشري.
iii. **قابلية القراءة (readability)**: أن يكون رماز اللغة التنفيذية الناتج عن عملية التحويل قابل للقراءة والفهم من قبل البشر.

إذ إنَّه في أغلب الأحيان يكون رماز اللغة التنفيذية الناتج عن التحويل غير قابل للقراءة من قبل البشر، وهذا الشرط مهم وذلك لأن رماز اللغة التنفيذية لإجرائية العمل الناتج عن هذه التحول قد يحتاج أحيانا إلى تنقية وتهذيب وخاصة أثناء عملية الاختبار والتنفيذ.

وعليه فإن الهدف الأساسي من هذا البحث هو الوصول إلى نموذج للانتقال من التمثيل الرمزي لنمذجة إجرائيات العمل إلى اللغة التنفيذية لإجرائيات العمل والتي تمثل توصيفاً دقيقاً لخدمات الويب، ومحاولة الوصول إلى نموذج قادر على تحويل أي تمثيل رمزي صحيح إلى لغة تنفيذية قابلة للتطبيق عن طريق أدوات اللغة التنفيذية المتاحة، وأن يكون رماز اللغة التنفيذية الناتج واضح ومقروء من قبل المستخدم الذي يحاول ببعض الأحيان تدقيق هذا الرماز ومقارنته مع مخطط التمثيل الرمزي وقد يحتاج لتعديله في أكثر الأحيان.

1.2. أسئلة البحث

1.2.1. ما هو الوضع الراهن للانتقال بين التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل؟

سيتم ضمن هذا البحث القيام بدراسة مرجعية لعملية الانتقال بين التمثيل الرمزي لإجرائيات العمل واللغة التنفيذية لإجرائيات العمل من خلال استعراض الخوارزميات والمنهجيات المستخدمة سابقاً للانتقال بين التمثيل الرمزي واللغة التنفيذية ومناقشة نقاط الضعف التي تعاني منها كل خوارزمية وكذلك مواطن القوة التي تتميز بها هذه الخوارزميات.

1.2.2. كيف يمكننا الانتقال من التمثيل الرمزي لإجرائيات العمل إلى اللغة التنفيذية لإجرائيات العمل؟

سيتم تطوير خوارزمية للانتقال بين التمثيلين مبنية على أساس تعديل خوارزمية إيشويس [16] والتي تقوم بتجميع الخدمات ضمن بنى مهيكلة يمكن تحويلها بسهولة إلى لغة تنفيذية لإجرائيات العمل، كما سيتم معالجة مشكلة الروابط المتزامنة التي تعاني منها هذه الخوارزمية.

1.2.3. هل يجب أن يحقق الدخل (التمثيل الرمزي لإجرائيات العمل) بعض الشروط؟

لوحَظَّ من خلال دراسة منهجيات الانتقال السابقة، وجود بعض الشروط المفروضة على نموذج الدخل كشرط أن يكون بيان الإجرائية مهيكل وغير حلقي [5]، وشرط أن كل بوابة تفريق يتبعها بوابة أو عدة بوابات تجميع من نفس النوع وكذلك يجب على الدخل أن يحقق شروط البناء الجيد لمخطط لإجرائية العمل [8]، وغيرها من الشروط المذكورة ضمن الدراسة المرجعية.

سيتم ضمن هذا العمل مناقشة هذه الشروط ودراسة إمكانية الاستغناء عنها أو عن بعضها، وأثر ذلك على عملية التحويل.

1.2.4. ماهي المعاير الذي يجب أن يحققها الخرج (اللغة التنفيذية لإجرائيات العمل)؟

يتم التركيز كثيرا على أن يكون رماز اللغة التنفيذية الناتج عن عملية التحويل قابل للقراءة والفهم من قبل المستخدم الذي يحاول ببعض الأحيان تدقيق هذا الرماز ومقارنته مع مخطط التمثيل الرمزي وقد يحتاج لتعديله في بعض الأحيان.

وسيتم ضمن هذا العمل بناء نموذج تحويل يكون خرجه رماز لغة تنفيذية قابل للقراءة بشكل جيد.

1.3. مقترح الحل

تنطلق فكرة الحل من خوارزمية إيشـــويس [16] التي تقوم بتجميع الخدمات ضـــمن إجرائيات مهيكلة[3]، فدخل هذه الخوارزمية هو بيان تبعيات للخدمات وخرجها شـجرة إجرائيات مهيكلة يمكن تحويلها بسـهولة إلى رماز لغة تنفيذية قابل للقراءة. لذلك سنقوم في البداية بتعديل هذه الخوارزمية لتقبل نموذج إجرائية عمل (BPMN) كدخل ويكون الخرج رماز لغة تنفيذية (BPEL)، ومن ثم سيتم تطوير هذه الخوارزمية لجعلها تقبل نماذج التمثيل الرمزي غير المهيكلة والتي تحتوي روابط متزامنة بين الفروع المتوازية، أي أننا نسعى لتحقيق معامل الكمال للخوارزمية من خلال تطويرها لتقبل نماذج دخل جديدة لا تقبلها الكثير من خوارزميات التحويل السابقة. كما ستتم برمجة وتنجيز الخوارزمية المعدَّلة باستخدام بيئة Eclipse بهدف تحقيق متطلب الأتمتة والتقليل من التدخل اليدوي بعملية التحويل.

1.4. فرضية البحث

نفترض أننا نسـتطيع الوصـول إلى نموذج للتحويل بين التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل، وأنَّ رماز اللغة التنفيذية الناتج عن عملية التحويل باسـتخدام النموذج قابل للقراءة البشـرية، وهذا التحويل قادر على تحويل جميع نماذج التمثيل الرمزي إلى اللغة التنفيذية.

1.5. مخطط البحث

قمنا بتنظيم هذا البحث ضمن عدة فصول، فقد بدأنا في الفصل الثاني بدراسة مرجعية تضمن شرحاً مبسطاً للتمثيل الرمزي واللغة التنفيذية وناقشـنا العديد من منهجيات التحويل بين النموذجين ونقاط ضـعف وقوة هذه المنهجيات، لننتقل بالفصـل الثالث لطرح نموذج الحل المقترح وتنجيزه برمجياً وتجربته على مثال بسـيط، ولنقوم ضـمن الفصـل الرابع بطرح حالتين دراسـيتين لتجربة النموذج المقترح عليها ومقارنة النتائج مع نتائج الأدوات البرمجيـة المتوفرة (BABEL, eclipse BPMN2BPEL plugin)، وليتم تقييم نموذج الحـل ضمن الفصل الخامس، ووضع الخاتمة والآفاق المستقبلية للحل ضمن الفصل السادس والأخير.

[3] الاجرائيات المهيكلة: هي الاجرائيات التي تملك تدفق تسلسلي محدّد ومعرَّف ويمكن تمثيله بقوالب تدفق نموذجية معرّوفة ومدّعومة من قبل أي أداة تمثيلية للإجرائيات، ويمكن تقسيم مخططات هذه الإجرائيات إلى مناطق ذات نقطة دخل واحدة ونقطة خرج واحدة [23].

الفصل الثاني

دراسة مرجعية للانتقال بين

التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل

2. دراسة مرجعية للانتقال بين التمثيل الرمزي واللغة التنفيذية

سيتم ضمن هذا القسم إجراء دراسة تفصيلية لكل من التمثيل الرمزي لإجرائيات العمل BPMN واللغة التنفيذية لإجرائية العمل BPEL، ومن ثم سيتم ذكر بعض المحاولات السابقة للانتقال بين التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل (BPMN to BPEL) والمقارنة بينها.

2.1. التمثيل الرمزي لنمذجة إجرائية العمل BPMN

كانت المحاولة الأولى للتمثيل الرمزي لإجرائيات العمل في عام 2004 وتمثلت بمبادرة نمذجة إجرائيات العمل (BPMN) بهدف تقديم تمثيل رمزي معياري لإجرائيات العمل، وأقرَّت منظمة الإدارة الغرضية (OMG) هذه المبادرة في عام 2006 كتوصيف للنمذجة البيانية لإجرائية العمل [1]. فقد كان الهدف من التمثيل الرمزي لإجرائيات العمل (BPMN) إيجاد رموز وتعابير مشتركة بين محللي ومطوري النظم لتمثيل إجرائيات العمل [1]، كما ساعد وجود هذه النمذجة البيانية في الانتقال من التمثيل البياني الصوري إلى اللغة التنفيذية (BPEL) والهدف منها هو تضييق الفجوة بين خبراء ومحللي إجرائيات العمل في أي مجال تجاري أو صناعي أو خدمي أو... وبين التقنيين المساعدين لهم.

تستخدم BPMN في نمذجة إجرائيات العمل للمؤسسة أو الشركة ولا تدعم نمذجة البنية التنظيمية للمؤسسة أو قواعد واستراتيجيات المنظومة.

يمكن تقسيم إجرائيات العمل من وجهة نظر نمذجة هذه الإجرائيات إلى ثلاثة أقسام:

1. **إجرائيات عمل خاصة (داخلية)**: توّصف الأنشطة الداخلية ضمن المنظومة، ولا يتم نشر هذه الإجرائيات خارج المنظومة وذلك للحفاظ على فعالية وخصوصية هذه الإجرائيات.
2. **إجرائيات عمل مختصرة عامة**: وهي إجرائيات عمل خاصة للمؤسسة، ولكن ينشر مختصر عنها للشركات المتعاملة معها، بهدف تسيير بعض العمليات التعاونية فيما بينها.
3. **إجرائيات عمل عامة شاملة**: يتم نشرها بشكل كامل وتفصيلي للعامة.

يتم التمثيل الرمزي لإجرائيات العمل باستخدام عناصر معيارية تشكل مخطط إجرائية العمل (BPD). فالمخطط يتكون من عناصرBPMN وهذه العناصر موضحة ضمن الشكل (2.1) وتتكون بشكل أساسي من أغراض ومن عناصر ربط ومسارات عمل [1].

- **الأغراض**: تتكون من أحداث ومهام وبوابات:
  - **الحدث Event**: **من** الممكن أن يكون:

    - إشارة لبدء إجرائية (start event)
    - إشارة لنهاية الإجرائية (end event)
    - إشارة الإنهاء الفوري للإجرائية (end terminate event)
    - رسالة تصل أو تحدد زمن الوصول أثناء الإجرائية (intermediate message/timer event)

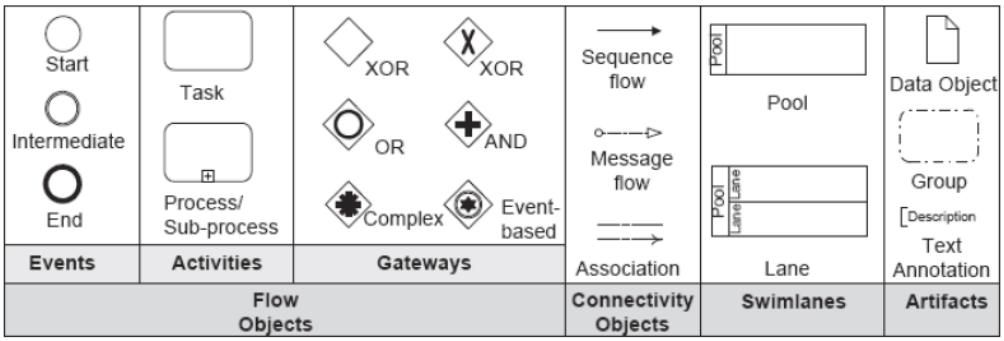
  - **المهمة**: هي نشاط أو عمل بسيط يجب أن ينجز خلال الإجرائية.
  - **البوابة**: هي بنية التوجيه التي تستخدم للسيطرة على التفريق والتجميع بين سلاسل التدفق، حيث يوجد:

- بوابة تفرع على التوازي (parallel fork gateways): تستخدم لخلق سلاسل تدفق متزامنة.
- بوابة جمع ع التوازي (parallel join gateways): تستخدم لتجميع سلاسل التدفق المتزامنة.
- بوابة XOR المبنية على البيانات: وتستخدم لاختيار أحد سلاسل التدفق المتوازية وذلك طبقا لبيانات الإجرائيات.
- بوابة XOR المبنية على الحدث: وتستخدم لاختيار أحد سلاسل التدفق المتوازية وذلك طبقا لحدث خارجي.
- بوابة XOR التجميعية: وتستخدم لتجميع عدة سلاسل تدفق ضمن سلسلة تدفق.

- **عناصر الربط: وهي**
  - سلاسل التدفق لإجرائيات العمل
  - خطوط تدفق الرسائل
  - خطوط المزامنة

- **مساري العمل (Swimlanes):**
  - الصندوق Pool: تحوي الإجرائيات الفرعية والمشاركين فيها لتشكيل الإجرائية الرئيسية.
  - المسارات Lines: تستخدم لتنظيم وتصنيف الأنشطة.

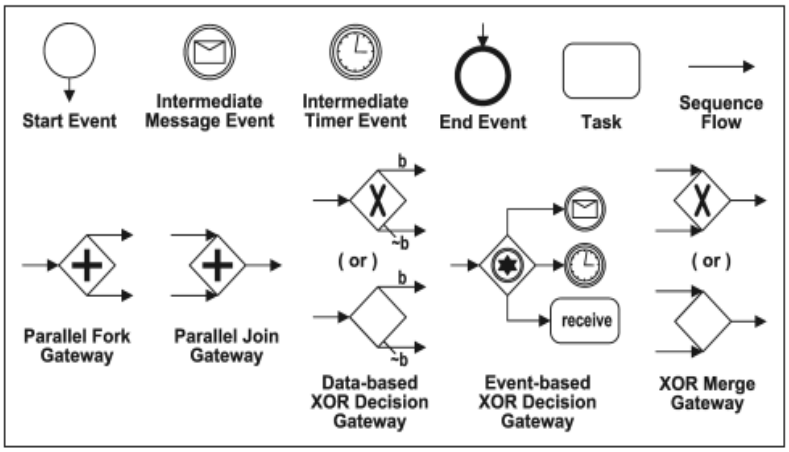
- **المصنوعات Artifacts**: ضمن التمثيل الرمزي يوجد 3 مصنوعات:
  - أغراض البيانات: لتوفير المعلومات اللازمة لتنفيذ الأنشطة ونتيجة التنفيذ.
  - المجموعات: تستخدم للتوثيق فقط وبيان الهدف من الإجرائية.
  - الحاشية الكتابية: تقدم معلومات إضافية لقراءة نماذج التمثيل الرمزي.



الشكل (2.1) العناصر الأساسية للتمثيل الرمزي لإجرائيات العمل

2.2. مخطط إجرائية العمل (BPD):

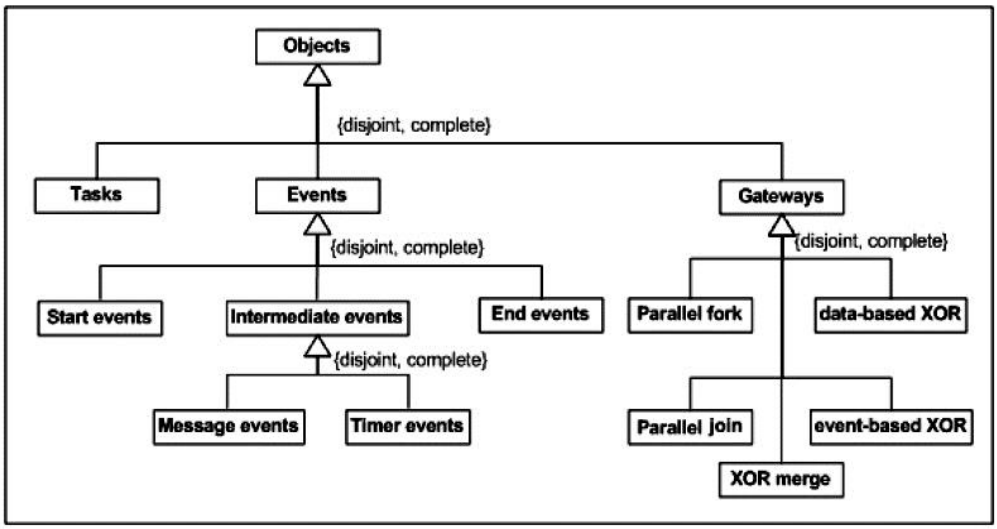
يتم استخدام مخطط إجرائية العمل للتعبير عن إجرائيات العمل ضمن التمثيل الرمزي لإجرائيات العمل، وهذا المخطط يتكون من عناصر BPMN الأساسية الموضحة في الشكل (2.2).



الشكل (2.2) عناصر مخطط إجرائية العمل وأنواعها

هذه العناصر تتكون كما رأينا سابقا بشكل أساسي من أغراض (O) والتي الممكن أن تكون مهمة (T) أو حدث (E) أو بوابة (G)، والأحداث تقسم إلى أحداث للبدء $\mathcal{E}^{\mathcal{S}}$ وأحداث وسيطة $\mathcal{E}^{\mathcal{I}}$ وأحداث للانتهاء $\mathcal{E}^{\mathcal{E}}$ ، وتقسم الأحداث الوسيطة إلى أحداث وسيطة رسائل $\mathcal{E}^{\mathcal{I}}_{\mathcal{M}}$ وأحداث وسيطة للمؤقت $\mathcal{E}^{\mathcal{I}}_{\mathcal{T}}$

أما البوابات فتتألف من بوابات للتفريق G^F وبوابات للتجميع G^J وبوابات القرار XOR المبني على البيانات G^D وبوابات القرار XOR المبنية على الأحداث G^V وبوابات XOR الدمج G^M. ويوضح الشكل (2.3) مخطط الصفوف للعناصر السابقة وعلاقتها ببعضها.



الشكل (2.3) مخطط الصفوف لعناصر التمثيل الرمزي وعلاقتها ببعضها

أما العلاقات بين الأغراض فتتم عن طريق علاقات تدفق التحكم $\mathcal{F} \subseteq \mathcal{O} \times \mathcal{O}$

العلاقة F تعرف بيان موجه يتألف من عقد (الأغراض) ووصلات (سلسلة تدفق). وهذا التعريف يسمح بوجود بيان غير مترابط (لا يملك أحداث بدء وخرج، أغراض بدون دخل أو خرج...). لكن بما أننا نريد الحصول على نموذج لا يحوي مشاكل كالقفل المميت والحلقات اللانهائية، لذلك فإننا نضطر للتعامل مع مخططات إجرائية عمل معرفة بشكل جيد، ومعظم خوارزميات التحويل (BPMN to BPEL) تطلب تحقق شروط البناء الجيد لمخطط إجرائية العمل المستخدم كدخل لهذه الخوارزميات.

تعريف1 : **مخطط إجرائية عمل معرف بشكل جيد**، فالعلاقة F تحقق مجموعة من المتطلبات [7]،[8]:

- $\forall\ s \in \mathcal{E}^{\mathcal{S}},\ in(s) = \emptyset \wedge |out(s)| = 1$, *i.e. start events have an indegree of zero and an outdegree of one,*

- $\forall\ e \in \mathcal{E}^{\mathcal{E}},\ out(e) = \emptyset \wedge |in(e)| = 1$, *i.e. end events have an outdegree of zero and an indegree of one,*

- $\forall\ x \in \mathcal{T} \cup \mathcal{E}^{\mathcal{I}},\ |in(x)| = 1$ *and* $|out(x)| = 1$, *i.e. tasks and intermediate events have an indegree of one and an outdegree of one,*

- $\forall\ g \in \mathcal{G}^{\mathcal{F}} \cup \mathcal{G}^{\mathcal{D}} \cup \mathcal{G}^{\mathcal{V}},\ |in(g)| = 1 \wedge |out(g)| > 1$, *i.e. fork or decision gateways have an indegree of one and an outdegree of more than one,*

- $\forall\ g \in \mathcal{G}^{\mathcal{J}} \cup \mathcal{G}^{\mathcal{M}},\ |out(g)| = 1 \wedge |in(g)| > 1$, *i.e. join or merge gateways have an outdegree of one and an indegree of more than one,*

- $\forall\ g \in \mathcal{G}^{\mathcal{V}},\ out(g) \subseteq \mathcal{E}^{\mathcal{I}} \cup \mathcal{T}^{\mathcal{R}}$, *i.e. event-based XOR decision gateways must be followed by intermediate events or receive tasks,*

- $\forall\ g \in \mathcal{G}^{\mathcal{D}},\ \exists x \in out(g),\ Cond((g,x)) = \neg\wedge_{y \in out(g)\setminus\{x\}} Cond((g,y))$, *i.e. (g,x) is the default flow among all the outgoing flows from g.*

- $\forall\ x \in \mathcal{O},\ \exists(s,e) \in \mathcal{E}^{\mathcal{S}} \times \mathcal{E}^{\mathcal{E}},\ s\mathcal{F}^{*}x \wedge x\mathcal{F}^{*}e$, *i.e. every object is on a path from a start event to an end event.*

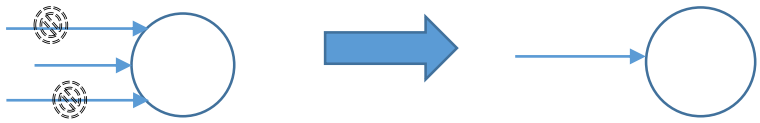
❖ أي إنه يتوجب على مخطط إجرائية العمل أن يحقق **الشروط التالية** ليكون **معرَّف بشكل جيد**:

i. حدث البداية يملك نقطة خرج واحدة ولا يملك نقطة دخل

فهو يحدد بداية الإجرائية، وبالتالي لا يجب أن يسبقه أي غرض، ومنه يتم اتخاذ مسار العمل للدخول إلى البوابات والأنشطة ومناقشة الشروط وتسلسل إجرائية العمل.

ii. حدث النهاية يملك نقطة دخل واحدة ولا يملك نقطة خرج

فهو يحدد نهاية الإجرائية، وبالتالي لا يجب أن يتبعه أي غرض ولا يمكن خروج منه أي رابط، حيث تتم معالجة منطق العمل كله قبل الوصول إليه ليتم الوصول إليه عن طريق رابط واحد فقط.

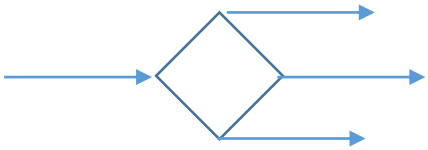


iii. المهام والاحداث الوسيطة تملك نقطة دخل واحدة ونقطة خرج واحدة

أي أنه لا يسمح بوجود أكثر من دخل أو أكثر من خرج للمهمة ضمن مخطط إجرائية العمل المعرَّف بشكل جيد، فهي ليست بوابة تفريق أو تجميع ليتم اتخاذ قرار عندها بتقسيم مسارات العمل أو تجميعها، وإنما يصلها مسار العمل ليتم عندها القيام بنشاط محدد فقط ويخرج نتيجة التنفيذ برابط واحد فقط أيضا. وكذلك لا يسمح بوجود مهمة بدون دخل أو خرج ويجب على كل مهمة أن تكون على أحد مسارات العمل كما سنجد في الشرط الأخير.

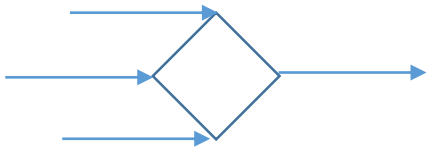
iv. بوابات القرار والتفريق تملك نقطة دخل واحدة وعدة نقاط للخرج:

يجب على بوابات القرار والتفريق أن تملك أكثر من نقطة خرج، فمهمتها تفريق مسار العمل لعدة مسارات متوازية أو اتخاذ القرار لتحديد مسار العمل وفق شرط محدد. أي يأتيها دخل واحد ويخرج منها أكثر من مسار يمكن تنفيذها كلها أو أحدها حسب نوع بوابة التفريق والقرار.



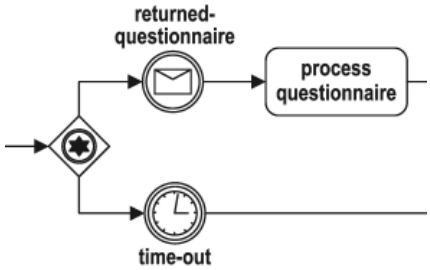
v. بوابات التجميع والدمج تملك عدة نقاط دخل ونقطة خرج واحدة

تقوم بتجميع عدة مسارات ضمن مسار واحد، فهي تملك عدة نقاط دخول ونقطة خرج واحدة.



vi. بوابة القرار XOR المبنية على الحدث يجب أن تتبع بأحداث وسيطة أو تستقبل مهام

يتوجب على بوابة القرار XOR المبنية على الحدث أن تُتبَع بحدث وسيط أو أن تتلقى مهمة وإلّا كانت بوابة مبنية على البيانات ويجب تصحيحها وتمثيلها كبوابة قرار مبنية على البيانات وليست بوابة مبنية على حدث.



vii. يجب أن يكون هناك مسار تدفق افتراضي لبوابات القرار المبني على البيانات

وذلك للتأكد من استمرارية تدفق إجرائية العمل في حالة عدم تحقق أي شرط من شروط الخرج الخاصة بالبوابة.

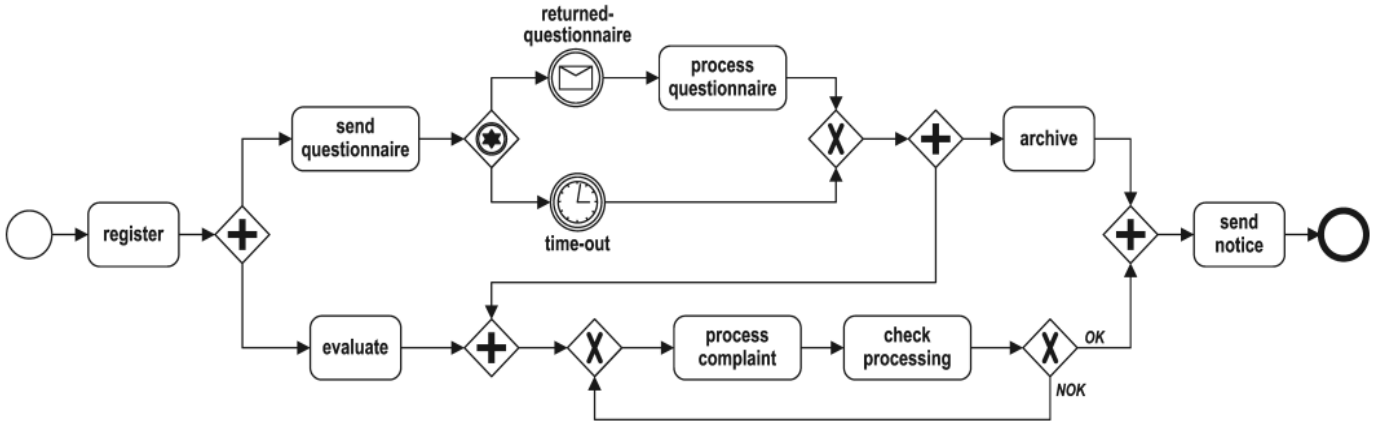
viii. كل غرض يجب أن يكون على مسار من حدث البداية وحتى حدث النهاية

يجب أن يكون مخطط إجرائية العمل صحيح ولا يحوي أي غرض معزول ليس له دور ضمن إجرائية العمل. أي لا بدَّ أن يكون هناك مسار يمكن أن يمر بهذا الغرض وإلا فإنَّ وجود هذا الغرض لا معنى له ويمكن أن يؤدي إلى مشاكل أثناء معالجة هذا المخطط، فالأغراض ضمن المخطط يجب أن تكون مرتبطة ببعضها ويجري المرور عليها وفقا لشروط وأحداث دخل ونتائج تنفيذ بعض المهام. فيتوجب الوصول لكل غرض من النموذج انطلاقا من حدث البداية وأيضا كل غرض يجب أن يكون على مسار لحدث نهاية وذلك لعدم الوقوع بمشكلة القفل المميت (deadlock).

❖ سيتم ضمن الفصل الخامس من هذه الأطروحة دراسة هذه الشروط بالتفصيل وأهمية وجود كل شرط من هذه الشروط بالنسبة للنموذج المقترح وكذلك سيتم مناقشة إمكانية التراخي عن بعض هذه الشروط وانعكاسه على نموذج الحل.

➢ مثال للتمثيل الرمزي لنمذجة إجرائية العمل BPMN:

الشكل (2.4) يوضح تمثيل رمزي لإجرائية العمل باستخدام BPMN تعبر عن إجرائية معالجة شكوى مقدمة إلى مؤسسة أو منظمة [6]، ففي البداية يتم تسجيل الشكوى (**task** register)، ليتم بعدها بالتوازي ارسال الاستطلاع (**task** send questionnaire) وتقييم الشكوى (**task** evaluate). إذا قام المدعى بالإجابة على الاستطلاع خلال أسبوعين (**event** returned questionnaire) يتم تنفيذ إجرائية معالجة الاستطلاع، وإلا (**event** time-out) فإنَّه سيتم تجاهل الاستطلاع، وفي كلتا الحالتين يتم ارشفة النتيجة (**task** archive). وبعد الانتهاء من تقييم الشكوى بشكل متوازي يتم البدء بالمعالجة الحقيقية للشكوى (**task** process complaint). ومن ثم يجري التحقق من معالجة الشكوى عن طريق (**task** check processing)، وإن كانت النتيجة سلبية فالشكوى بحاجة إلى إعادة معالجة، أما في حالة النتيجة الإيجابية فيتم ارشفة الاستطلاع. يتم إرسال إشعار لإبلاغ صاحب الشكوى عند الانتهاء من معالجة الشكوى (**task** send notice).



الشكل (2.4) مخطط إجرائية عمل لمعالجة شكوى

2.3. اللغة التنفيذية لإجرائية العمل BPEL

تعد منظمة النهوض بمعايير المعلومات المنظمة (OASIS) المنظمة الداعمة للغة التنفيذية لإجرائيات العمل (BPEL) [2]، حيث تقود هذه المنظمة عملية تطوير وتعريف المعايير الخاصـــة بهذه اللغة، كما طوّرت هذه المنظمة عدة معايير خاصة بخدمات الويب مثل (SOAP, WSDL , UDDI).

تقوم اللغة التنفيذية لإجرائيات العمل بتوصـــيف احتمالات التفاعل بين خدمات الويب وترتيب هذه الخدمات في إجرائية العمل، ويوجد ضـــمن توصـــيف اللغة التنفيذية عدة طرق لتطبيق مفاهيمها فيمكن تطبيقها **كلغة توصـــيفية** لإجرائيات العمل لا تهدف للتنفيذ كما يمكن تطبيقها **كلغة تنفيذية** وهنا يجب بنائها بشكل دقيق لتكون قابلة للتنفيذ. فنحو اللغة التنفيذية مبني بشكل جيد وعناصره متداخلة بشكل منظم، وكل عنصر يمكن أن يكون له موصفات (attribute) ويمكن أن يحوي عناصر أبناء متداخلة معه. فاللغة التنفيذية لإجرائيات العمل تحتوي العديد من العناصر التي تستخدم لغرض توصيف وبناء إجرائيات العمل.

يمكن تقسيم عناصر اللغة التنفيذية لإجرائيات العمل ضمن قسمين أساسيين: الأنشطة (activities) وروابط الشـركاء (partner links) [2]. فالأنشـطة في منطق اللغة التنفيذية هي التي تنجز الإجرائيات (process)، وهذه الأنشـطة تنقسـم إلى أنشـطة أسـاسـية (basic activities) وأنشـطة مهيكلة (structured activities). فالأنشطة الأساسية توصف الخطوات الأساسية لعمل الإجرائية بينما الأنشطة المهيكلة توصف منطق التحكم بتدفق الأنشطة وهذه الأنشطة مفصلة ضمن الجدول (2.1). أما روابط الشركاء فتسمح بتفعيل حالة (instance) من خدمات الويب التي نتخاطب معها، فهي تقوم بتوصيف التفاعل بين إجرائيات عمل الشركات المتعاونة عن طريق خدمات الويب.

الأنشطة الأساسية	
الاستدعاء invok	تستخدم لاستدعاء العمليات من خدمات الويب الشريكة
الاستقبال والرد Receive and Reply	إجرائية تنتظر رسالة محددة وترد عليها
التعيين Assign	من أجل تغيير المتحولات
الرمي وإعادة الرمي Throw and Rethrow	لمعالجة الأخطاء، كل العناصر يمكن أن تلقي أخطاء لمعالج الأخطاء
الانتظار Wait	عندما تحتاج إجرائية العمل إلى التأخر مدة من الزمن
النشاط الفارغ Empty	لا تنفذ شيء، لكن تستخدم لتساعد في توليد رماز محقق وقابل للقراءة
الخروج Exit	إنهاء جميع الأنشطة في إجرائية العمل

الأنشطة المهيكلة	
لتسلسل Sequence	الأنشطة المتسلسلة التي يتم تنفيذها واحدة تلو الأخرة في ترتيب معين
الشرط If	للشرط
مادام، كرر حتى While & repeatUntil	تكرار مجموعة من الانشطة
إلتقاط، اختيار Pick	ينتظر لحدوث حدث محدد ومن ثم تنفيذ الأنشطة المتعلقة بهذا الحدث
التدفق Flow	الأنشطة المتسلسلة التي يتم تنفيذها على التوازي
في حالة كل نشاط ForEach	لمعالجة الإجرائية ذات الفروع المتعددة اعتمادا على تصميم هذا النشاط، يمكن تنفيذ أبناء النشاط على التوالي أو بالتوازي

الجدول (2.1) توضيح لعناصر اللغة التنفيذية لإجرائيات العمل

➢ مثال عن اللغة التنفيذية لإجرائية العمل BPEL

يمثل الرماز التالي رماز اللغة التنفيذية لإجرائية العمل الموافق لمخطط معالجة الشكوى المذكورة في القسم السابق:

```
<process>
    <links>
      <link name="t3TOt4"/>
    </links>
    <sequence name="t_c^8">
      <invoke name="register"/>
      <flow name="t_c^5">
        <sequence name="t_c^6">
          <sequence name="t_c^3">
            <source linkName="t3TOt4"/>
            <invoke name="send questionnaire"/>
            <pick name="t_c^2">
              <onMessage name="returned-questionnaire">
                <invoke name="process questionnaire"/>
              </onMessage>
              <onAlarm name="time-out">
                <empty/>
              </onAlarm>
            </pick>
          </sequence>
          <invoke name="archive"/>
        </sequence>
        <sequence name="t_c^7">
          <invoke name="evaluate"/>
          <sequence name="t_c^4">
            <target linkName="t3TOt4"/>
            <sequence name="t_c^1">
              <invoke name="process complaint"/>
              <invoke name="check processing"/>
            </sequence>
             <while condition="NOK">
               <sequence name="t_c^1">
                 <invoke name="process complaint"/>
                 <invoke name="check processing"/>
               </sequence>
             </while>
           </sequence>
         </sequence>
       </flow>
       <invoke name="send notice"/>
     </sequence>
  </process>
```

2.4. الأبحاث السابقة للتحويل بين التمثيل الرمزي واللغة التنفيذية لإجرائية العمل

عمل الكثير من الباحثين سابقا على تطوير خوارزمية لإجراء التحويل من التمثيل الرمزي لنمذجة إجرائيات العمل BPMN إلى اللغة التنفيذية لإجرائيات العمل BPEL، وتم وضع عدة منهجيات للانتقال السابق. وسنناقش في هذا القسم الاختلاف بين التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل وكذلك سنذكر الأبحاث السابقة للتحويل بين النموذجين السابقين، وسيكون **الكمال** (إمكانية تحويل أي تمثيل رمزي إلى لغة تنفيذية) و**قابلية القراءة** هما المعياران الاساسيان في مناقشة منهجيات التحويل.

2.4.1. الاختلاف بين التمثيل الرمزي واللغة التنفيذية لإجرائية العمل BPMN,BPEL

يمكن تلخيص الاختلاف بين التمثيل الرمزي واللغة التنفيذية لإجرائية العمل ضمن الفروقات الرئيسية التالية:

i. تُستخدَم اللغتان في **مرحلتين مختلفين ضمن مرحلة تطوير وبناء إجرائية العمل**، فالتمثيل الرمزي لنمذجة إجرائية العمل يستخدم في مرحلة التصميم حيث يقوم محللو الاعمال بمناقشة وتصميم إجرائية العمل باستخدام الرموز البيانية، بينما يجري استخدام اللغة التنفيذية لإجرائية العمل في مرحلة التنفيذ حيث يعمل التقنيون والمبرمجون الذين يستخدمون اللغات البرمجية التنفيذية.

ii. الاختلاف بين محللي الاعمال والتقنيين الذين ينظرون إلى اللغتين بشكل مختلف.

iii. الطبيعة المختلفة للنموذجين فالتمثيل الرمزي لإجرائيات العمل هي لغة بيانية تستخدم الرموز والبيانيات، بينما اللغة التنفيذية فهي لغة كتلية مهيكلة.

وهذا يؤدي بالضرورة لوجود فجوة مفاهيم كبيرة بين النموذجين تشكل مشكلة حقيقية، تعمل أدوات التحويل BPMN to BPEL على تضييقها وحلها. ويمكن أن يساعد محللو الاعمال ومهندسو البرمجيات على فهم إجرائية العمل بشكل أكبر عند اطلاعهم على كل من نموذجي اللغة التنفيذية والتمثيل الرمزي لإجرائية العمل نفسها.

قام ريكر ومينـدلينغ [4] بإجراء مقارنة بين التمثيل الرمزي لنمذجة الإجرائيات واللغة التنفيذية لإجرائيات العمل من حيث دعمهما لقوالب التدفق (workflow patterns) وكانت النتائج الموضحة ضمن الجدول (2.2). حيث تشير الإشارة (+) إلى دعم اللغة لنموذج التدفق والإشارة (−) إلى عدم دعم اللغة للنموذج بينما الإشارة (+/−) تدل على دعم اللغة للنموذج بشكل غير مباشر.

Workflow Patterns	BPMN	BPEL	Workflow Patterns (ctd.)	BPMN	BPEL
Basic Control Flow			11. Implicit Termination	+	+
1. Sequence	+	+	*Multiple Instances Patterns*		
2. Parallel Split	+	+	12. MI without Synchronization	+	+
3. Synchronization	+	+	13. MI with a priori Design Time Knowledge	+	+
4. Exclusive Choice	+	+	14. MI with a priori Runtime Knowledge	+	–
5. Simple Merge	+	+	15. MI without a priori Runtime Knowledge	–	–
Adv. Synchronization			*State-Based Patterns*		
6. Multiple Choice	+	+	16. Deferred Choice	+	+
7. Synchronizing Merge	+/–	+	17. Interleaved Parallel Routing	+/–	+/–
8. Multiple Merge	+	–	18. Milestone	–	–
9. Discriminator	+	–	*Cancellation Patterns*		
Structural Patterns			19. Cancel Activity	+	+
10. Arbitrary Cycles	+	–	20. Cancel Case	+	+

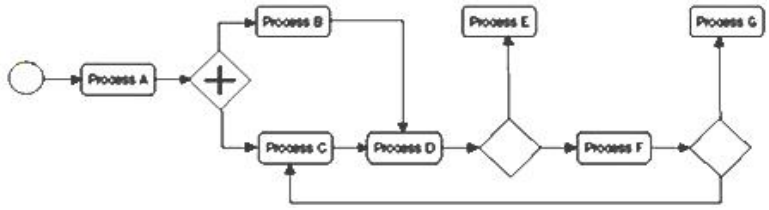
الجدول (2.2) المقارنة بين التمثيل الرمزي واللغة التنفيذية من حيث دعم نماذج التدفق

- **القوالب المدعومة من قبل التمثيل الرمزي وغير مدعومة من قبل اللغة التنفيذية**

i. **الدمج المتعدد (Multiple merge)**: التجميع غير المتزامن لفرعين متباعدين أو أكثر. إذا كان أكثر من فرع فعّال (نشط)، فإن نشاط الدمج يبدأ مع كل تنشيط لكل فرع قادم.

ii. **Discriminator**: تجميع فرعين أو أكثر بفرع تالي أخر، بحيث يكون تنشيط الفرع التالي عند أول نتيجة تصل من أي من الفروع الداخلة ويجري اهمال الفروع الأخرى.

iii. **حلقات كيفية Arbitrary cycles**: لتمثيل الحلقات ذات نقط الدخول والخروج المتعددة.



حيث يمكن الدخول إلى الحلقة عند الإجرائية (C) وكذلك عند الإجرائية (D).

iv. **الاســـتدعاء المتعدد مع أولوية معرفة زمن التشـــغيل MI with a priori runtime knowledge**: القدرة على البدء بعدة استدعاءات للنشاط الواحد ضمن الاستدعاء نفسه للإجرائية. قد يختلف عدد الاستدعاءات في كل مرة لكن يجب معرفته عند بداية الاستدعاء. يبدأ تنشيط النشاط اللاحق عند اكتمال تنفيذ كامل الاستدعاءات.

❖ كما يظهر الجدول (2.2) فإنّ القوالب الأربعة الســـابقة مدعومة من قبل التمثيل الرمزي وغير مدعومة من قبل اللغة التنفيذية لإجرائيات العمل. إن هذا الاختلاف بدعم القوالب يؤثر بالتأكيد على عملية التحويل والانتقال بين اللغتين، ولكن هذا التأثير يعتمد على الهدف من التحويل وكذلك مخطط التمثيل الرمزي لإجرائية العمل المراد تحويله إلى لغة تنفيذية من حيث احتوائه على واحد أو أكثر من النماذج الســـابقة. فمن المحتمل أنّ يكون مخطط الدخل لا يحتوي أي من القوالب غير المدعومة من قبل اللغة التنفيذية وبالتالي هذا الاختلاف بالدعم لا يؤثر على مســـار التحويل. لذلك يجب أن يكون محللو الاعمال الذين يرغبون بتحويل مخططاتهم إلى اللغة التنفيذية على علم مسبق بهذا الاختلاف بالدعم.

2.4.2. آخر الأبحاث في مجال التحويل بين التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل

يناقش ميندلينغ [5] عملية التحويل والانتقال بين نوعين مختلفين من لغات نمذجة الإجرائيات: اللغات ذات التوجه الكتلي (BPEL, BPML) واللغات ذات التوجه البيانية (BPMN, EPCs,). إذا تطرح خمس منهجيات لتحويل بيان الإجرائية (Process Graphs) إلى اللغة التنفيذية لإجرائيات العمل، ومن ثم تناقش عدة طرائق للتحويل العكسي أي الانتقال من اللغة التنفيذية إلى الشكل البياني. فهذا التحويل هام بالنسبة لبعض الباحثين بالهندسة العكسية، والمنهجيات هي: الحفاظ على العنصر (element preservation)، تصغير العنصر (Element minimization)، تطابق البنية (Structure identification)، توسيع البنية (Structure maximization)، قواعد حدث-شرط-فعل (Event-condition-action-rules).

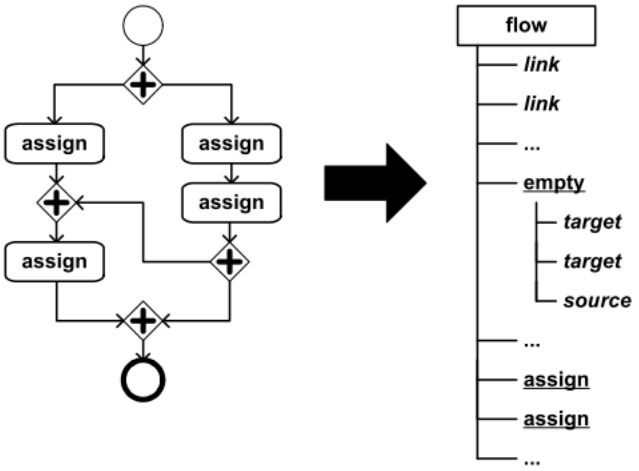
i. **منهجية الحفاظ على العنصر**

تقوم هذه المنهجية بما يلي:

a. تحويل كافة عناصر بيان الإجرائية إلى تدفق: elements → flow

b. تحويل الروابط إلى عناصر فارغة: connectors → <empty> elements

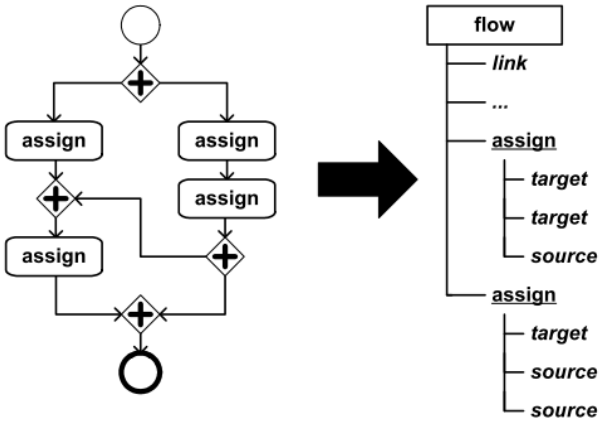
c. تحويل الأقواس إلى روابط: arcs → <links>



الشكل (2.5) تمثيل منهجية الحفاظ على العنصر

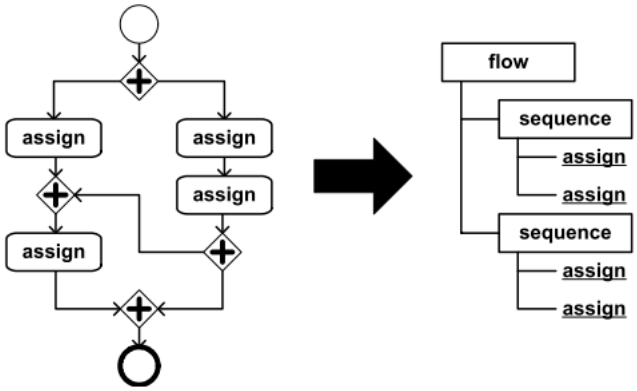
تتميز هذه المنهجية بسهولة تنفيذها والتقابل المتين (واحد-واحد) بين عقد البيان البدائي للإجرائية ورماز اللغة التنفيذية، **إلا أنّ** رماز اللغة التنفيذية الناتج يحتوي عناصر أكثر بالإضافة إلى صعوبة فهم التدفق ضمن اللغة. لذلك يجب عدم استخدام هذه المنهجية من أجل المخططات الكبيرة والتي يكون معيار قابلية القراءة فيها مهماً، كما أنَّ هذه المنهجية لا تصلح لتحويل مخططات التمثيل الرمزي التي تحتوي حلقات.

ii. **منهجية تصغير العنصر[5]**: وهي امتداد لمنهجية الحفاظ على العنصر حيث يتم حذف العناصر الفارغة، فتستخدم هذه المنهجية للتقليل من العقد الناتجة، مما ينعكس بشكل أفضل على الأداء وقابلية القراءة.



الشكل (2.6) تمثيل لخوارزمية تصغير العنصر

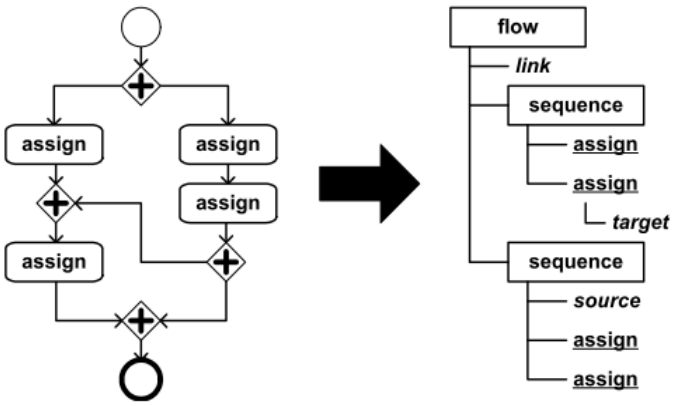
iii. **منهجية تطابق البنية[5]**: تستخدم هذا المنهجية عندما تكون الإجرائية مهيكلة بشكل جيد، حيث يجري تضمين الإجرائية بمكون واحد خلال عملية التحويل. وتتميز هذه المنهجية بأن كل تدفقات التحكم يجري تحويلها إلى أنشطة مهيكلة برماز لغة تنفيذية قابل للقراءة.



الشكل (2.7) تمثيل لخوارزمية تطابق البنية

iv. **منهجية توسيع البنية[5]**: تعتبر هذه المنهجية توسيعاً للمنهجية السابقة (تطابق البنية)، بحيث يمكن إجراء التحويل على بيان الإجرائيات غير المهيكلة بشرط ألا تحتوي على حلقات كيفية (اعتباطية). حيث يتم استخدام منهجية الحفاظ على العنصر ومنهجية تصغير العنصر من اجل أجزاء إجرائية العمل غير المهيكلة والتي لا يمكن استخدام منهجية تطابق البنية معها.

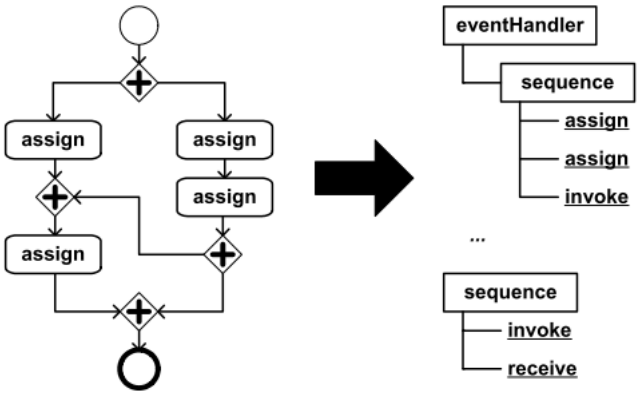
- ومن مساوئ هذه المنهجية الصعوبة الناتجة عن تطبيق عدة استراتيجيات أثناء التنفيذ للوصول إلى الحل.



الشكل (2.8) تمثيل لخوارزمية توسيع البنية

v. **منهجية قواعد حدث–شرط–فعل[5]**: هذه المنهجية مبنية أيضا على منهجية تطابق البنية. إذ يتم تحويل الأجزاء المهيكلة من الإجرائية باستخدام منهجية تطابق البنية بينما يتم تحويل الأجزاء الأخرى باستخدام معالجات الأحداث.

تتميز هذه المنهجية بقدرتها على تحويل معظم بيانات إجرائيات العمل بما فيها البيانات التي تحتوي حلقات غير مهيكلة، لكن مشكلة هذه المنهجية أن رماز اللغة التنفيذية لإجرائيات العمل الناتج عن عملية التحويل غير قابل للقراءة بسبب استخدام معالجات الأحداث.

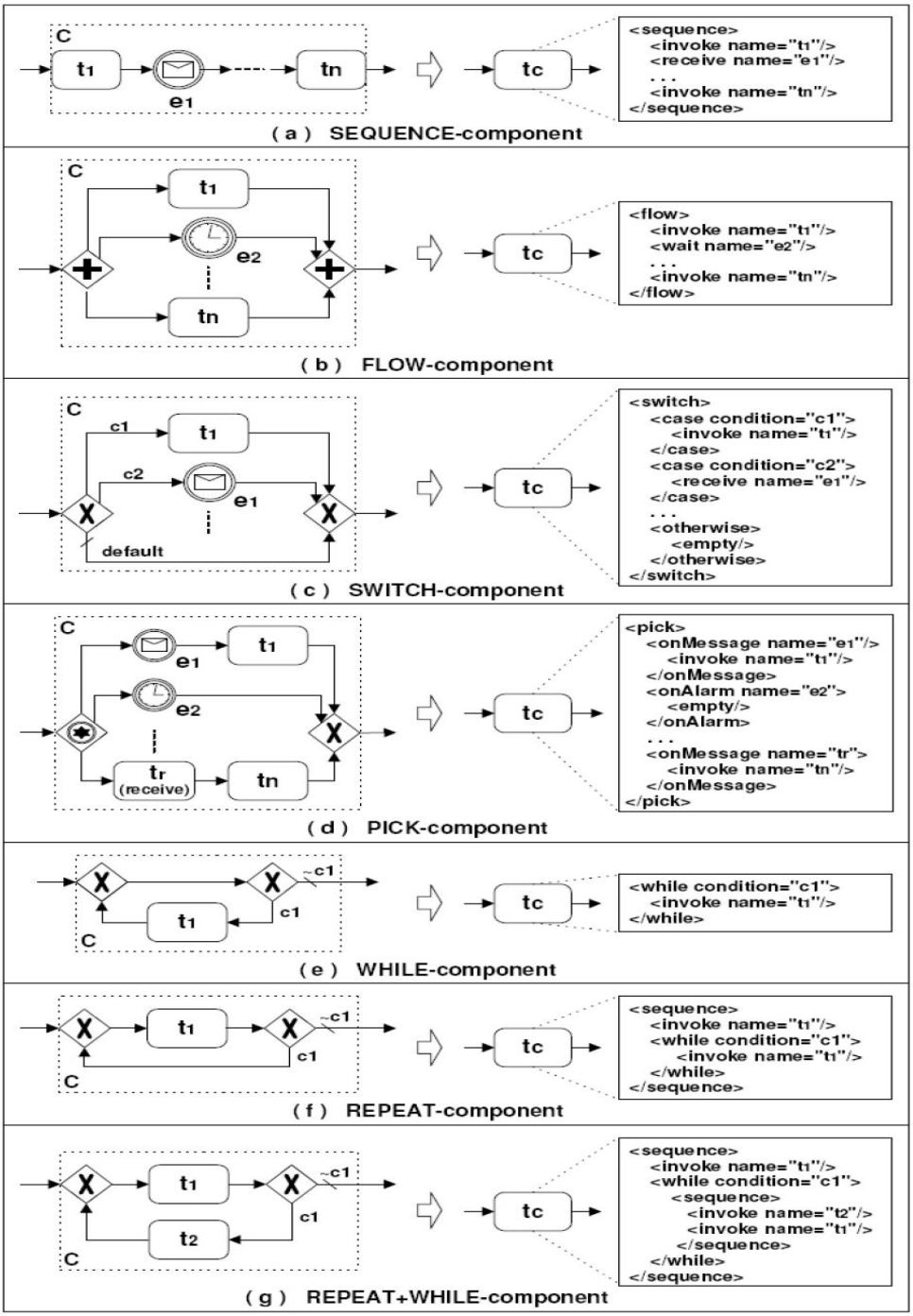


الشكل (2.9) تمثيل لخوارزمية قواعد حدث–شرط–فعل

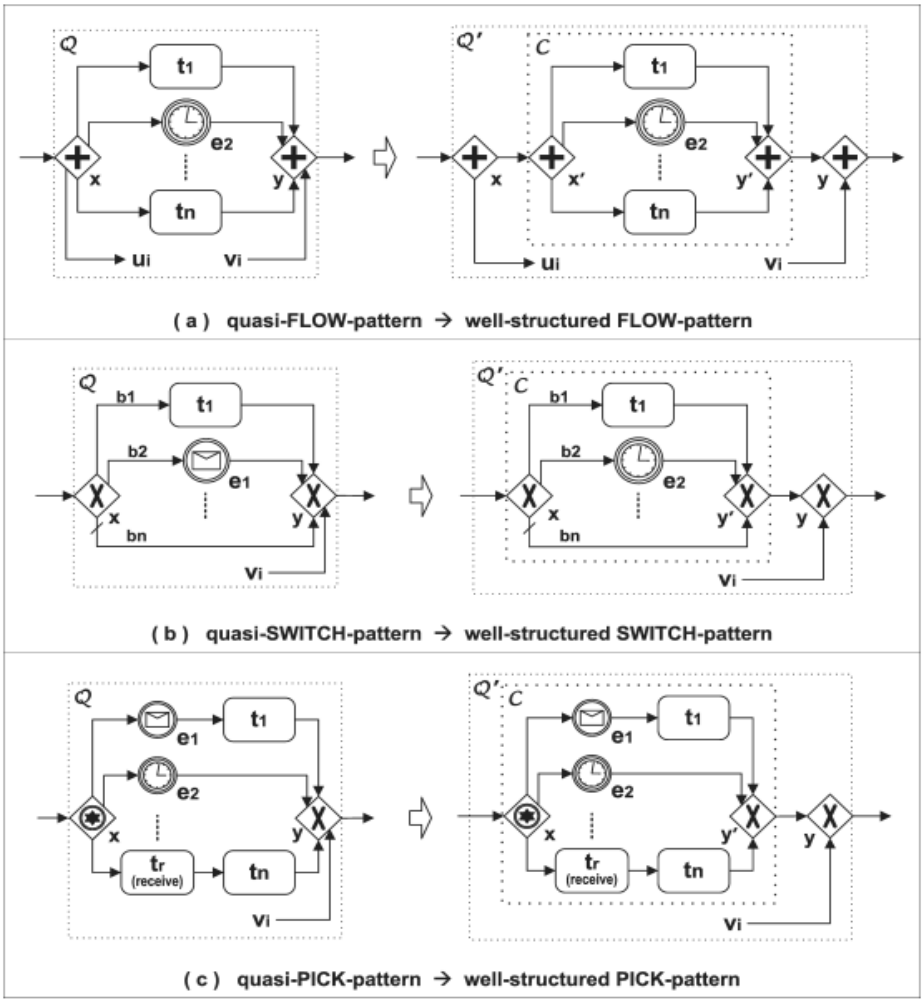
- يطرح أويان [6] خوارزمية تحويل مبنية على قاعدة حدث–فعل، إذ تبدأ الخوارزمية بتفكيك مخطط إجرائية العمل(BPD) إلى مكوناته، بحيث يتم تحويل المكونات المهيكلة بشكل جيد باستخدام منهجية **تطابق البنية** المذكورة سابقاً، بينما تستخدم **قاعدة حدث–فعل** بتحويل المكونات الأخرى. ففي البداية يتم تعريف الشروط المسبقة التي تنتج قواعد حدث–فعل ليتم بعدها تحويل هذه القواعد إلى رماز اللغة التنفيذية لإجرائيات العمل. والشكل (2.10) يوضح طريقة تحويل النماذج المبنية بشكل جيد إلى رماز اللغة التنفيذية الموافق لها.

تابع أويان عمله [7] من خلال تطوير خوارزميته السابقة [6] المبنية على قاعدة حدث–فعل وذلك من خلال استخدام شبكات بيتري (Petri nets) للتأكد من صحة وسلامة مخطط إجرائية العمل (BPD) قبل تطبيق خوارزمية قاعدة حدث–فعل. كما تم تمثيل التبعية والعلاقة بين الأنشطة كروابط تحكم (control links)، وذلك بهدف جعل رماز اللغة التنفيذية الناتج أكثر قابلية للقراءة حيث أن روابط التحكم يجري تحويلها إلى تدفق التحكم ضمن رماز اللغة التنفيذية الناتج.

كذلك تابع أويان تطوير عمله [8] حيث عمل على تحويل النماذج شبه المهيكلة (Quasi-structured patterns) إلى نماذج مهيكلة مطابقة للقوالب ضمن الشكل (2.10). أي أنَّه قام بزيادة عدد القوالب التي تستخدم فيها منهجية مطابقة البنية في عملية التحويل وقللَّ من استخدام قواعد حدث–فعل مما انعكس إيجاباً على قابلية قراءة رماز الخرج.

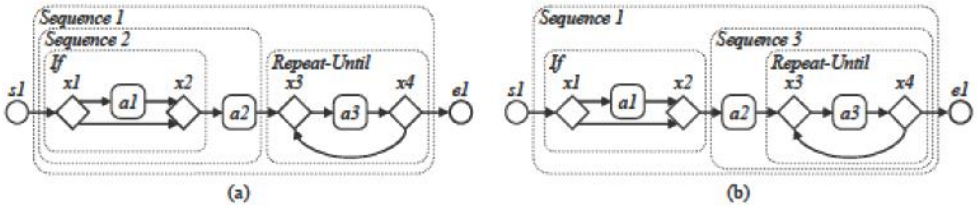


الشكل (2.10) تحويل النماذج المبنية بشكل جيد إلى رماز اللغة التنفيذية



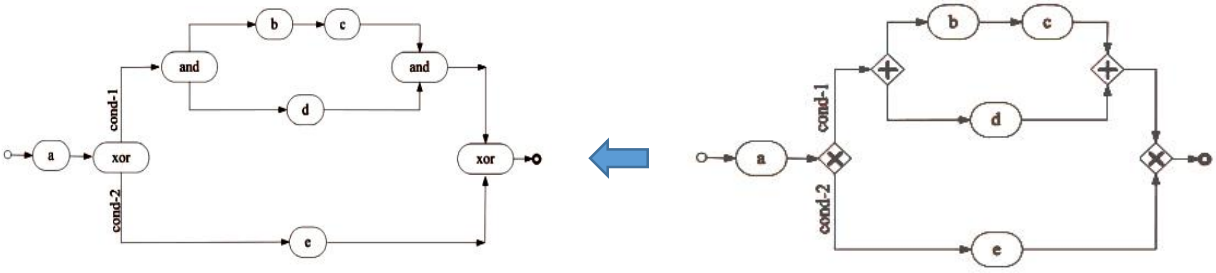
الشكل (2.11) تحويل النماذج شبه المهيكلة إلى نماذج مهيكلة ضمن منهجية أويان

وتســتخدم هذه الطريقة في التحويل من أجل النماذج الحتمية فقط وإلا فإن إجراء التحويل على نفس الدخل من التمثيل الرمزي لإجرائيات العمل قد يعطي رماز خرج مختلف من اللغة التنفيذية، وذلك نتيجة إنه يمكن أن يجري تجزئة نفس النموذج بأشـــكال مختلفة، ليجري تجميعها بعد التحويل بطريقة مختلفة.



- يطرح وايت [9] منهجية ترتكز على عنصر التدفق (Flow Element) ضمن اللغة التنفيذية، ويتم ضمنها تحويل التمثيل الرمزي لنمذجة الإجرائيات إلى بنية بيانية للغة التنفيذية بدلا من البنية الكتلية (عنصر تسلسلي). لذلك فعناصر الربط لا تستخدم لأغراض التزامن فقط وإنما للعلاقات الأخرى بين الأنشطة. ويمكن تطبيق هذه الإجرائية لتحويل نماذج التمثيل الرمزي غير المهيكلة، لكن رماز اللغة التنفيذية الناتج يفتقر لقابلية القراءة.

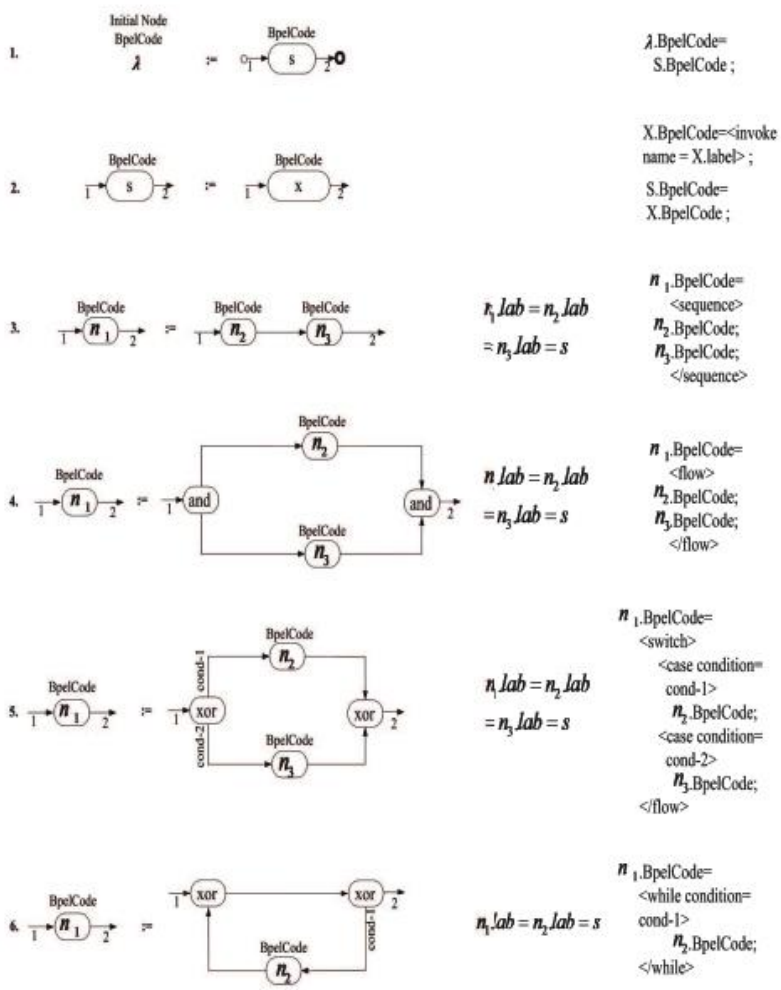
- تقوم منهجية قواعد البيان [10] على تحويل التمثيل الرمزي إلى اللغة التنفيذية باستخدام قواعد البيان. فيتم في البداية تحويل مخطط إجرائية العمل إلى بيان بحيث تُمثَّل بوابات ومهام مخطط إجرائية العمل بعُقَّد ضمن البيان بينما تُمثَّل الروابط بوصلات البيان كما في الشكل (2.11)، ومن ثم يجري مطابقة عناصر البيان السابق بنماذج صنعية – الشكل (2.12). وهذه العناصر تمثل أجزاء من بيان تمت إضافة معامل إلى عقدها لتوصيف رماز اللغة التنفيذية الذي تمثله هذه العقدة. فينتج لدينا بيان عقده توصف عناصر مخطط إجرائية العمل مع رماز اللغة التنفيذية المقابل لها – الشكل (2.13).



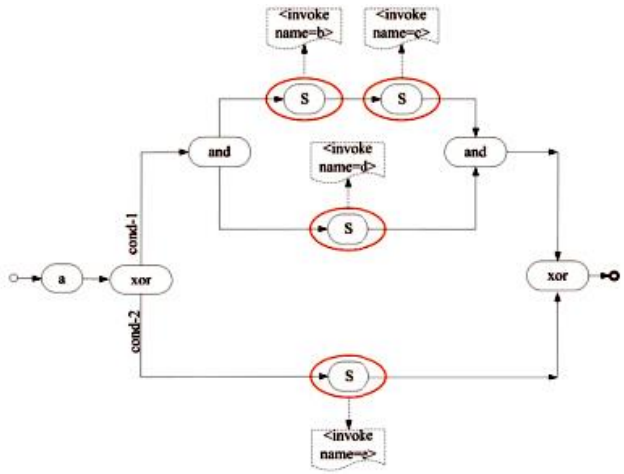
الشكل (2.12) تحويل مخطط إجرائية إلى بيان ضمن خوارزمية تشي–زينغ

يتم بعد ذلك تجميع رماز اللغة التنفيذية من خلال تجميع عقد البيان والرماز الممثل لها بخطوات متتالة كما في الشكل (2.14) لنحصل بالنهاية على بيان يتألف من عقدة واحدة ورماز اللغة التنفيذية المرافق لها يمثل رماز اللغة التنفيذية لإجرائية العمل كاملة.

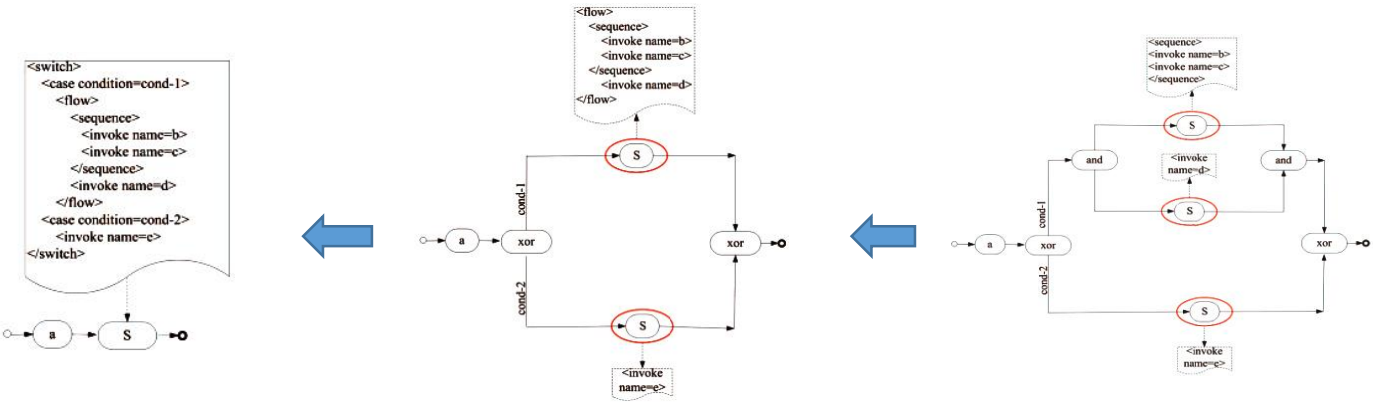
تقوم هذه الخوارزمية بتحويل مكونات مخطط إجرائية العمل المهيكلة فقط لذلك فهي تعالج فقط الحلقات المهيكلة ذات نقطة الدخول الواحدة ونقطة الخروج الواحدة ولا تعالج الحلقات غير المهيكلة ذات نقاط الدخول أو الخروج المتعددة.



الشكل (2.13) نماذج صنعية توصف عناصر BPMN الأساسية ضمن البيان E-EGGs



الشكل (2.14) بيان وسيط للانتقال إلى اللغة التنفيذية ضمن خوارزمية تشي-زينغ



الشكل (2.15) مراحل دمج عقد البيان للوصول إلى رماز اللغة التنفيذية ضمن خوارزمية تشي-زينغ

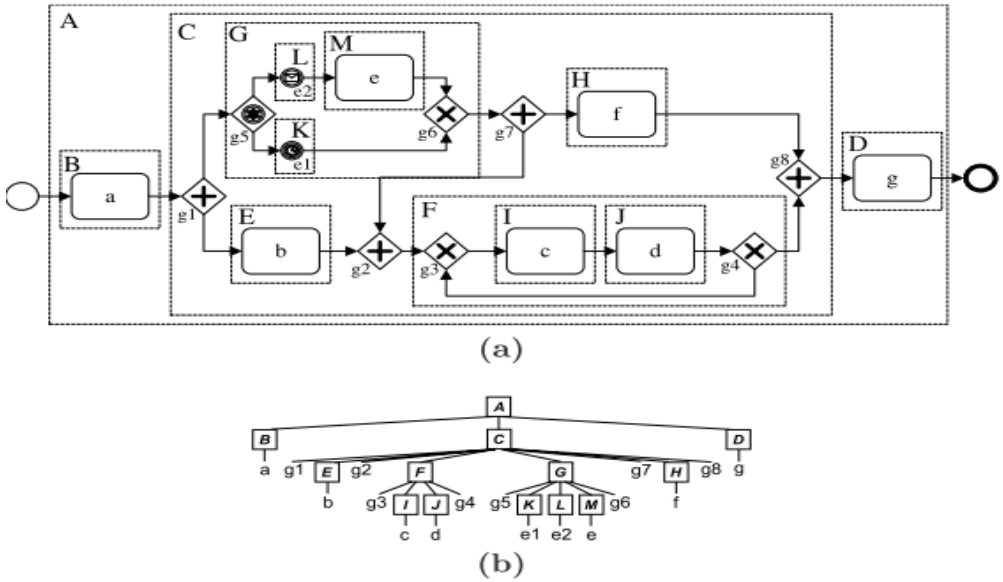
- تعتمد منهجية تصـــنيف وتطابق النماذج [15] على تفكيك مخطط إجرائية العمل إلى أجزاء ومن ثم تحويل هذه الأجزاء إلى اللغة التنفيذية لإجرائيات العمل. أي أن هذه المنهجية تتألف من خطوتين أساسيتين. يتم ضمن الخطوة الأولى تحليل مخطط إجرائية العمل لتحديد مناطق ذات نقطة دخل واحدة ونقطة خرج واحدة (SESE)، هذه المناطق تم تعريفها ضـــمن [6]، ومن ثم يتم تجميع المناطق الســابقة بنفس ترتيبها وطريقة تداخلها لتشكل شجرة تسمى شجرة الإجرائية المهيكلة (Process Structure Tree)، لتتم ضـــمن الخطوة الثانية عملية تحليل لتدفقات التحكم لكل منطقة لجمع المعلومات وتحديد النموذج (Pattern) الموافق ضمن كل مجموعة وتحويله إلى اللغة التنفيذية لإجرائيات العمل.

تعريف2 : **مناطق ذات نقطة دخل واحدة – نفطة خرج واحدة (SESE–region)** [6]:

Let BPD = (O, F, Cond) be a well-formed core BPD. C = (O_c,F_c , $Cond_c$) is a component (SESE-region) of BPD if and only if:

- $\mathcal{O}_c \subseteq \mathcal{O} \backslash (\mathcal{E}^S \cup \mathcal{E}^E)$,
- $|\ (\cup_{x \in \mathcal{O}_c}\ in(x)) \backslash \mathcal{O}_c\ | = 1$, *i.e. there is a single entry point outside the component, which can be denoted as* $entry(C) = elt((\cup_{x \in \mathcal{O}_c}\ in(x)) \backslash \mathcal{O}_c)$,
- $|\ (\cup_{x \in \mathcal{O}_c}\ out(x)) \backslash \mathcal{O}_c\ | = 1$, *i.e. there is a single exit point outside the component, which can be denoted as* $exit(C) = elt((\cup_{x \in \mathcal{O}_c}\ out(x)) \backslash \mathcal{O}_c)$,
- *there exists a unique source object* $i_c \in \mathcal{O}_c$ *and a unique sink object* $o_c \in \mathcal{O}_c$ *and* $i_c \neq o_c$, *such that* $entry(C) \in in(i_c)$ *and* $exit(C) \in out(o_c)$,
- $\mathcal{F}_c = \mathcal{F} \cap (\mathcal{O}_c \times \mathcal{O}_c)$,
- $Cond_c = Cond(\mathcal{F}_c)$, *i.e. the Cond function where the domain is restricted to* $\mathcal{F}_c$.

وكما هو واضـــح من التعريف فهذه المناطق تملك نقطة دخل واحدة ونقطة خرج واحدة والعلاقات المعرفة ضـــمن هذه المناطق هي الروابط بين أغراض هذه المناطق فقط، والتابع Cond[4] يقتصـــر على العلاقات السابقة فقط.



الشكل (2.16) تحويل مناطق (SESE) إلى شجرة الإجرائية المهيكلة

[4] التابع Cond: هو تابع لتحويل التدفقات المتسلسلة إلى شروط تأخذ القيمة true أو false إذا تم تنفيذ التدفق المتسلسل أم لم يتم التنفيذ

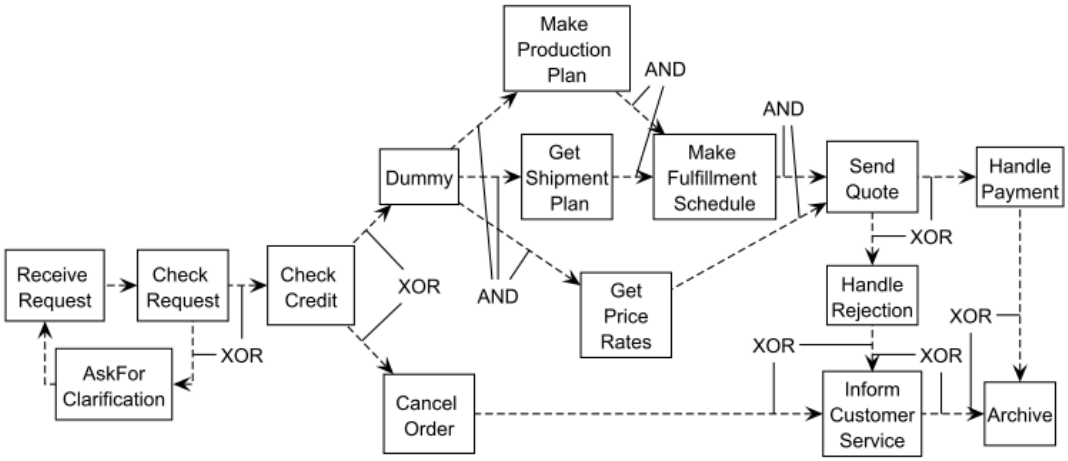
وقد تم بناء إضـــافة برمجية BPMN2BPEL على بيئة Eclipse تســـتند على هذه المنهجية، وتقوم هذه الإضـــافة بعملية التحويل بين التمثيل الرمزي واللغة التنفيذية وهذه الأداة متوفرة على الرابط [12]، وتتميز هذه الأداة بقدرتها على تحويل جميع مخططات إجرائيات العمل إلى اللغة التنفيذية، ولكن تعاني هذه المنهجية من صعوبة فهم رماز اللغة التنفيذية الناتج.

- يقدم إيشـــويس [16] خوارزمية لتجميع مجموعة من الخدمات وتبعياتهم[5] ضـــمن إجرائيات مهيكلة يمكن بسـهولة تحويلها إلى اللغة التنفيذية لإجرائيات العمل. تأخذ هذه الخوارزمية بيان التبعيات كدخل وتنتج نموذج مهيكل للإجرائيات يعبر عن علاقة هذه الإجرائيات وترابطها فيما بينها. هذا النموذج الممكن تحويله بسـهوله إلى لغة تنفيذية لإجرائيات العمل. حيث يتم ضـمن هذه المنهجية تحليل تدفقات الدخل والخرج بين الخدمات وتحويلها مباشـــرة إلى تدفق تحكم لنحصـــل بالنتيجة على تراكيب مبنية على البيان. يتم تنســـيق وتنظيم الخدمات باستخدام عناصر التحكم (AND-joins XOR-forks, XOR-joins,AND-forks).

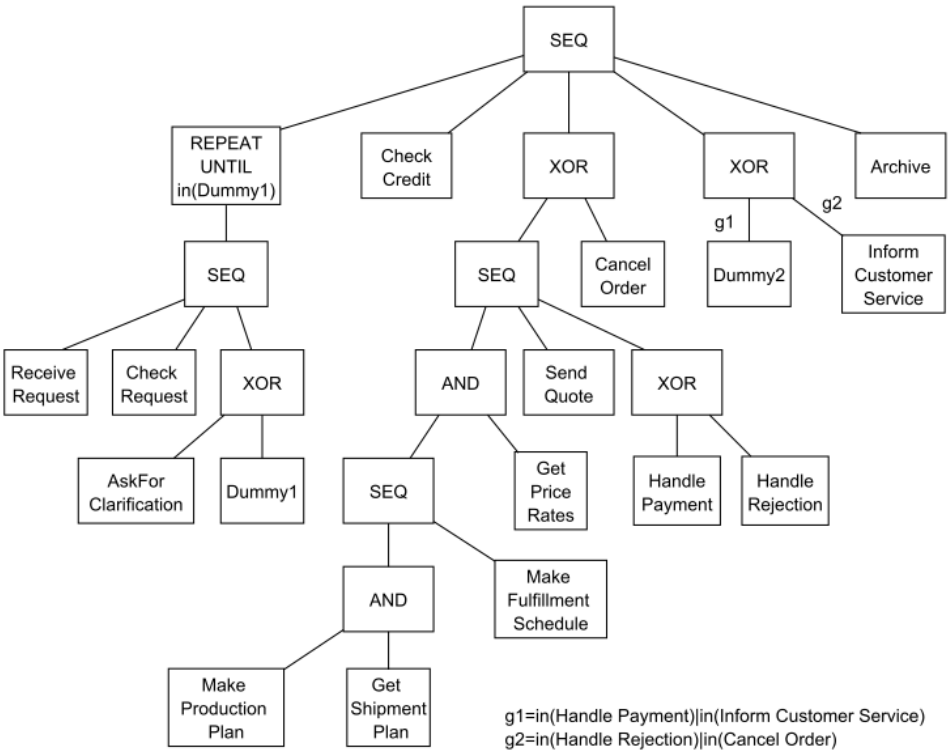
يوضـح الشـكل (2.16) بيان التبعية الخاص بإجرائية طلب نشـرة أسـعار منتجات من شـركة، وإجراء عملية دفع ثمن طلبية منتجات من الشـــركات، والتي تتكون من مجموعة من الخدمات والتبعيات فيما بينها والتي يعبر عنها بأســـهم منقطة، فمثلا الخدمة (Receive Request) تعتمد على الدخل القادم من الخدمة (Check Request)، وفي حالة كانت الخدمة التي تملك أكثر من تبعية دخل أو خرج فإن الخدمات الســـابقة أو اللاحقة يجري تنفيذها جميعها (AND) أو تنفيذ واحدة فقط (XOR)، ويتم التعبير عن الشروط بإضافة خدمات فارغة (Dummy).

يتم معالجة بيان التبعيات السـابق باسـتخدام خوارزمية التجميع لنحصـل على التركيب المهيكل الموضـح في الشـــكل (2.17)، والتي يعبر عنها بشـــجرة أوراقها تمثل الخدمات والعقد الداخلية تمثل الشـــكل الكتلي للإجرائيات، فكل كتلة توصف شرط محدد.

[5] تبعيات الخدمات: الروابط التي تعبر عن العلاقة بين هذه الخدمات والتي يتم اشتقاقها من تدفق البيانات بين الخدمات، ويمكن أن تعبر أيضا عن الأحداث المتبادلة بين هذه الخدمات وكذلك الشروط التي تفرضها طبيعة الاعمال التي تمثلها هذه الخدمات



الشكل (2.17) بيان التبعية لإجرائية طلب أسعار ودفع فاتورة



الشكل (2.18) التركيب المهيكل الناتج عن تطبيق خوارزمية التجميع

إن الخوارزمية السابقة تقوم بتحويل بيان التبعية غير المهيكل إلى شجرة تراكيب مهيكلة يجري تحويلها بسهولة إلى اللغة التنفيذية لإجرائيات العمل برماز خرج قابل للقراءة بشكل كبير باعتبار أننا نقوم بتحويل بنى مهيكلة ومضبوطة.

تعاني هذه الخوارزمية مثل معظم خوارزميات تحويل الإجرائيات غير المهيكلة إلى إجرائيات المهيكلة من مشاكل ناتجة عن وجود حلقات كيفية أو وجود روابط متزامنة بين فروع متوازية ناتجة عن تزامنات سابقة. تضعف هذه الروابط قدرة الخوارزمية على تحويل الإجرائيات غير المهيكلة إلى إجرائيات مهيكلة [20]، [21]، [22].

2.4.3. دراسة نقدية

المشكلة الأساسية في التحويل بين التمثيل الرمزي واللغة التنفيذية هو أن دخل التحويل أي نماذج التمثيل الرمزي تكون مصممة بطريقة غير دقيقة، وتتضمن الكثير من المشاكل فقد يتكون النموذج من عقد يمكن أن تكون مرتبطة من خلال علاقات مسرودة بشكل عائم بدون قيود صارمة مما يؤدي في بعض الأحيان بالحصول على نماذج بدلالات تنفيذية غير مرغوبة قد تؤدي إلى مشاكل عديدة كالقفل المميت (deadlocks) أو الحلقات اللانهائية (livelocks). فهذه المشاكل التقنية لا تؤخذ بعين الاعتبار من قبل محللي النظم الذين يقومون بكتابة وصياغة سيناريوهات وإجرائيات العمل باستخدام التمثيل الرمزي لإجرائيات العمل، لذلك كان لابد من التأكد من سلامة وصحة نماذج التمثيل الرمزي قبل البدء بتحويله إلى اللغة التنفيذية لإجرائيات العمل. ومن أجل ذلك يقوم داماس [13] بطرح خوارزمية لتحويل التمثيل الرمزي لإجرائيات العمل إلى شبكات بيتري (Petri nets) والتي تساعدنا بالتأكد من سلامة وصحة نماذج التمثيل الرمزي قبل الشروع بعملية التحويل.

2.4.4. نقاط القوة والضعف لمنهجيات التحويل السابقة

بعد استعراض العديد من الخوارزميات والمنهجيات المتبعة في تحويل التمثيل الرمزي لإجرائيات العمل إلى اللغة التنفيذية لإجرائيات العمل، سنقوم بتحديد نقاط القوة والضعف لكل منهجية ودرجة كمالها من حيث قبولها كافة نماذج التمثيل الرمزي والشروط المطلوبة على هذه النماذج لنتمكن من تطبيق هذه المنهجية، والجدول (2.3) يوضح هذه المعلومات.

منهجية حفظ العنصر [5]	
نقاط القوة	- سهولة التطبيق. - سهولة المقارنة والمطابقة بين الدخل والخرج لأن كل من اللغة التنفيذية الناتجة وبيان الإجرائية الأساسية مبنيان على شكل بيان.
نقاط الضعف	صعوبة قراءة اللغة التنفيذية الناتجة وذلك بسبب كثرة عدد العناصر وروابط التحكم.
الشروط	يجب ألا يكون بيان الإجرائية حلقي.
الكمال	لا يمكن تطبيقها على كافة النماذج.
منهجية حذف العنصر [5]	
نقاط القوة	أداء جيد باعتبارها تحتوي العناصر الضرورية فقط
نقاط الضعف	صعوبة قراءة اللغة التنفيذية الناتجة ومقارنتها بالبيان الأصلي بسبب خاصية حذف العناصر
الشروط	يجب ألا يكون بيان الإجرائية حلقي.
الكمال	لا يمكن تطبيقها على كافة النماذج.
منهجية تطابق البنية [5]	
نقاط القوة	سهولة فهم رماز اللغة التنفيذية الناتج كونه يجري تحويل كامل بيان الإجرائية إلى أنشطة مهيكلة
نقاط الضعف	صعوبة مقارنتها مع بيان الإجرائية الأصلي بسبب أن رماز اللغة التنفيذية لا يعتمد بنية كتلية block-structured.
الشروط	يجب أن يكون بيان الإجرائية مهيكل.
الكمال	تقوم بتحويل الإجرائيات المهيكلة فقط ولا تدعم الإجرائيات غير المهيكلة.

منهجية توسيع البنية [5]	
نقاط القوة	تدعم الإجرائيات غير المهيكلة
نقاط الضعف	صعوبة التطبيق بسبب الحاجة لتطبيق عدة استراتيجيات أثناء التنفيذ للوصول إلى النتيجة.
الشروط	الحلقات في الإجرائيات غير المهيكلة يجب أن تحقق شرط نقطة دخل واحدة – نقطة خرج واحدة SESE
الكمال	تقوم تقريبا بتحويل جميع الإجرائيات ماعدا التي تحتوي حلقات اعتباطية (كيفية)
منهجية قواعد حدث–شرط–فعل[5]	
نقاط القوة	تدعم جميع الإجرائيات بما فيها التي تحوي حلقات غير مهيكلة.
نقاط الضعف	رماز اللغة التنفيذية الناتج غير مفهوم وصعب القراءة.
الشروط	لا يوجد شروط على الدخل
الكمال	تقوم بتحويل جميع الإجرائيات
التحويل BPMN to BPEL [8],[7]	
نقاط القوة	- تدعم الحلقات الاعتباطية (الكيفية) باستخدام التحويل المبني على الاحداث. - تناقش صحة وسلامة النماذج باستخدام شبكات بيتري. - تعالج بوابات التوازي والXOR. - رماز اللغة التنفيذية الناتج قابل للقراءة إلى حد ما، لإنه يتم تحويل المكونات المهيكلة قبل استخدام التحويل المبني على التدفق.
نقاط الضعف	- تســـتخدم نموذج الاحداث من اجل الإجرائيات غير المهيكلة مما يقلل من قابلية القراءة - صعوبة التطبيق باعتبارها تجمع بين ثلاثة طرق.
الشروط	- نماذج التمثيل الرمزي يجب أن تحقق شروط البناء الجيد لمخطط إجرائية العملBPD - يجب التحقق من كون نماذج التمثيل خالية من القفل المميت، وفي كثير من الأحيان يصعب التحقق من ذلك.

	- التحويل باستخدام روابط التحكم بالرغم من أن هذه الروابط من الممكن ألا تؤدي إلى حلقات ولا تحوي بوابات معتمدة على الاحداث ويجب أن تكون أمنة وسليمة
الكمال	قدرتها على تحويل كافة مكونات نموذج التمثيل الرمزي، فيتم في البداية تحويل المكونات المبنية بشكل جيد باستخدام منهجية تطابق البنية، وبعدها تستخدم الطريقة المبنية على الاحداث والطريقة المبنية على الروابط لتحويل باقي المكونات.
طريقة بنية البيان للغة التنفيذية [9]	
نقاط القوة	تحافظ على بنية البيان المستخدمة في نموذج التمثيل الرمزي لإجرائيات العمل
نقاط الضعف	صعوبة فهم رماز اللغة التنفيذية الناتج عن التحويل
الشروط	ضرورة توفر معلومات عن المتحولات ومجالاتها وكذلك الإجرائية العامة، فعادة هذه المعلومات لا تتوفر في جميع نماذج BPMN
الكمال	لم يتم التطرق إلى كافة مفاهيم BPMN ضمن المقالة، ولكن يمكن الحصول على درجة عالية من الكمال بشرط تحقيق شروط <link>-constructs
طريقة قواعد البيان [10]	
نقاط القوة	رماز اللغة التنفيذية الناتج مفهوم، وسهولة تطبيق الخوارزمية
نقاط الضعف	تعالج فقط المكونات المهيكلة
الشروط	مخطط إجرائية العمل يحوي فقط المكونات المهيكلة.
الكمال	لم يتم التطرق إلى كافة مفاهيم BPMN ضمن المقالة، ولا تعالج إلا المكونات المهيكلة.
نموذج شبكات بيتري [14]	
نقاط القوة	تملك شبكات بتري أساس نظري متين، وبالتالي أذا كانت شبكات بتري الداخلة آمنة وسليمة فإننا نضمن حصولنا في الخرج على رماز لغة تنفيذية صحيح.
نقاط الضعف	تتطلب عملية التحويل إلى شبكات بتري في البداية
الشروط	يجب أن تكون شيكات الدخل أمنة وسليمة، أي لها مدخل واحد ومخرج واحد وكل عقدة موجودة على مسار واحد من المدخل إلى المخرج.
الكمال	التحويل بين التمثيل الرمزي لإجرائيات العمل وشبكات بيتري غير كامل.

تصنيف وتطابق النماذج [15]	
نقاط القوة	القدرة على تحويل كافة نماذج التمثيل الرمزي بما فيها النماذج غير المهيكلة
نقاط الضعف	رماز اللغة التنفيذية الناتج غير مفهوم وصعب القراءة
الشروط	لا يوجد شروط على الدخل
الكمال	تقوم بتحويل جميع الإجرائيات
تجميع الخدمات ضمن تراكيب مهيكلة [16]	
نقاط القوة	القدرة على تحويل الخرج إلى رماز لغة تنفيذية قابلة للقراءة
نقاط الضعف	دخل الخوارزمية بيان تبعيات وخرجها شـــجرة إجرائيات، وبالتالي تحتاج إلى تعديلات ليصبح دخلها BPMN وخرجها BPEL.
الشروط	- كل بوابة تفريق يتبعها بوابة تجميع أو عدة بوابات تجميع من نفس النوع. - كل حلقة لها نقطة دخل وحيدة ونقطة خرج وحيدة من النوع xor.
الكمال	- لا تدعم الحلقات الاعتباطية. - تعاني من مشاكل مع الروابط المتزامنة بين الفروع المتوازية الناتجة عن تزامنات سابقة

الجدول 2.3 الاستراتيجيات الحالية المستخدمة لعملية التحويل بين التمثيل الرمزي واللغة التنفيذية

- نستطيع تلخيص الجدول السابق بجدول المقارنات التالي: حيث أنّ
  - أ- منهجية حفظ العنصر [5]
  - ب- منهجية تصغير العنصر [5]
  - ت- منهجية تطابق البنية [5]
  - ث- منهجية توسيع البنية [5]
  - ج- منهجية قواعد حدث-شرط-فعل [5]
  - ح- النموذج الهجين للتحويل BPMN to BPEL [8],[7]
  - خ- نموذج بنية البيان [9]
  - د- منهجية قواعد البيان [10]
  - ذ- تصنيف وتطابق النماذج [15]
  - ر- تجميع الخدمات ضمن إجرائيات مهيكلة [16]

حيث يتم استخدام الإشارة (+) للدلالة على دعم المنهجية للخاصية الموافقة والاشارة (−) لعدم دعمها، أما الإشـــارة (+/−) فتدل على الدعم الجزئي للخاصـــية من قبل المنهجية فقد تدعم المنهجية بعض مكونات الخاصية ببعض نماذج التمثيل الرمزي ولا تدعمها بنماذج أخرى.

	أ	ب	ت	ث	ج
قابلية قراءة الكود الناتج	**غير مفهوم** بسبب كثرة عدد العناصر وروابط التحكم	**صعوبة قراءة الكود** ومقارنته بالبيان الأصلي بسبب خاصية حذف العناصر	**سهولة فهم الكود** كونه يجري تحويل كامل بيان الإجرائية إلى أنشطة مهيكلة	**سهولة فهم الكود** بشرط عدم وجود مكونات غير مهيكلة	**غير مفهوم** بسبب استخدام قواعد حدث فعل
الكمال	**غير محقق** فالمنهجية لا تدعم الحلقات المهيكلة ولا الحلقات الغير مهيكلة	**غير محقق** فالمنهجية لا تدعم الحلقات المهيكلة ولا الحلقات الغير مهيكلة	**غير محقق** فالمنهجية لا تدعم المكونات والحلقات الغير مهيكلة	**مقبول إلى حد ما**، لكنها لا تدعم الحلقات الغير مهيكلة	يمكن اعتباره **محقق** مع تحفظ بالنسبة **لبعض** الروابط المتزامنة
المكونات المهيكلة	+	+	+	+	+
الحلقات المهيكلة	−	−	+	+	+
المكونات الغير مهيكلة	+	+	−	+	+
الحلقات الغير مهيكلة	−	−	−	−	+
الروابط المتزامنة	+	+	−	−/+	−/+

الجدول 2.4 مختصر المقارنة بين الاستراتيجيات المستخدمة لعملية التحويل بين التمثيل الرمزي واللغة التنفيذية

	ح	خ	د	ذ	ر
قابلية قراءة الكود الناتج	**قابل للقراءة إلى حد ما**، لأنه يتم تحويل المكونات المهيكلة قبل استخدام التحويل المبني على التدفق	**صعوبة قراءة الكود** المبني على أساس عنصر التدفق بدلا من عنصر التسلسل	**سهولة فهم الكود** كونه يعتمد على تجميع الكود بنفس هيكلية البيان	**غير مفهوم** بسبب استخدام قواعد حدث فعل	**مفهوم** باعتباره يقوم بتحويل فقط المكونات المهيكلة
الكمال	**مقبول إلى حد ما**، لكنها لا تدعم الروابط المتزامنة	**غير محقق** فالمنهجية لا تدعم الحلقات المهيكلة ولا الحلقات الغير مهيكلة	**غير محقق** فالمنهجية تدعم فقط المكونات المهيكلة	يمكن اعتباره **محقق** مع تحفظ بالنسبة **لبعض** الروابط المتزامنة	**غير محقق** فالمنهجية تدعم فقط المكونات المهيكلة
المكونات المهيكلة	+	+	+	+	+
الحلقات المهيكلة	+	−	+	+	+
المكونات الغير مهيكلة	+	+	−	+	−/+
الحلقات الغير مهيكلة	+	−	−	+	−
الروابط المتزامنة	−	+	−	−/+	−

متابعة الجدول 2.4 مختصر المقارنة بين الاستراتيجيات المستخدمة لعملية التحويل بين التمثيل الرمزي واللغة التنفيذية

2.4.5. الأدوات التنفيذية للتحويل المتوفرة حالياً:

يوجد أداتان مفتوحتا المصـــدر لإجراء عملية التحويل بين التمثيل الرمزي لنمذجة إجرائيات العمل واللغة التنفيذية لإجرائيات العمل هما (**BABEL**-tool, Eclipse **BPMN2BPEL** plug in).

- **BABEL** هي أفضـل الأدوات المتوفرة للتحويل وقد تم بنائها اعتمادا على المقالة [7]، إلا أن هذه الأداة لم يجرِ عليها أي تطويرات منذ إنشـــائها وهي تحتوي على الكثير من المشـــاكل والعيوب، بالإضـــافة إلى قلة التوثيق لهذه الأداة، كما أن نموذج التحليل نقطة دخل واحدة – نقطة خرج واحدة (SESE) غير الحتمي غير منفذ ضمنها.

- **Eclipse BPMN2BPEL** : وهي تطبيق يستند إلى تحليل شجرة بنية الإجرائية وقد تم بناء إضافة برمجية (***BPMN2BPEL plug in***) لبيئة التطوير Eclipse وهي متوفرة على موقع أكواد جوجل [12] وهذه الإضـافة مبنية اسـتنادا إلى المنهجية المطروحة ضـمن [15] وهي تسـتخدم نموذج التحليل نقطة دخل واحدة – نقطة خرج واحدة (SESE) وذلك من أجل تحديد مكونات الإجرائيات وبنيتها، وهذه البنى متداخلة ومتراكبة لتشكل في النهاية شجرة البنية، ومن ثم تستخدم الأداة لتحويل كل ورقة من أوراق الشجرة إلى بنية اللغة التنفيذية الموافقة لهذه الورقة.

- كلا البرمجيتين تعتمد على فكرة دمج أكثر من منهجية لإجراء التحويل بين التمثيل الرمزي واللغة التنفيذية، ففي البداية يتم تحديد المكونات المهيكلة وتحويلها ومن ثم يجري تحويل العناصـر التي يمكن تحويلها باسـتخدام المنهج المستند على الرابط، وفي النهاية يتم استخدام منهج حدث–فعل.

2.5. الخاتمة

قمنا باستعراض العديد من المنهجيات المستخدمة للتحويل بين التمثيل الرمزي لإجرائيات العمل واللغة التنفيذية، وقد تم مناقشة ميزات ومساوئ كل منها. حيث لاحظنا أن المعيارين الأساسيين للمقارنة وتقييم كفاءة هذه المنهجيات المختلفة للتحويل هما **الكمال** (القدرة على تحويل كافة نماذج التمثيل الرمزي إلى اللغة التنفيذية) و**قابلية قراءة** رماز اللغة التنفيذية الناتج، فالمنهجية المثلى هي المنهجية التي تستطيع أن تحول أكبر قدر ممكن من نماذج التمثيل الرمزي إلى رماز لغة تنفيذية قابل للقراءة البشرية، إلا إنه في كثير من الأحيان وجدنا أن هناك تعارض بإمكانية الوصول للوضع المثالي بالنسبة للمعيارين، حيث إنه كلما زاد كمال المنهجية لتحويل نماذج التمثيل الرمزي، كان رماز اللغة التنفيذية الناتج عن التحويل غير قابل للقراءة.

هناك اختلاف بطريقة تحويل الإجرائيات والمكونات المهيكلة وغير المهيكلة، فالمكونات المهيكلة يتم تحويلها إلى بنى للغة تنفيذية مترابطة تدعم قابلية القراءة بشكل جيد. أما المكونات غير المهيكلة فيتم تحويلها إلى روابط تحكم ومعالجات أحداث والتي يؤدي وجودها في رماز اللغة التنفيذية الناتج إلى إضعاف قابلية قراءته. إلا أن دعم الخوارزمية للبنى غير المهيكلة يرفع قدرتها لتحويل عدد أكبر بكثير من نماذج التمثيل الرمزي، بينما القيود على نماذج الدخل كشرط أن يكون مخطط إجرائية الدخل مكون بشكل جيد فهي تزيد من قابلية قراءة رماز اللغة التنفيذية الناتج. لذلك فإننا يمكن أن نحصل على خوارزمية تمتلك خاصية مقبولة من الكمال من ناحية نماذج الدخل ويكون رماز اللغة التنفيذية للخرج قابل للقراءة بشكل كبير وذلك من خلال دمج عدة منهجيات مع بعضها، بحيث نحصل على ميزات هذه الاستراتيجيات ونتجنب مساوئها.

الفصل الثالث

نموذج الحل المقترح

للانتقال بين

التمثيل الرمزي واللغة التنفيذية

لإجرائيات العمل

3. نموذج الحل المقترح

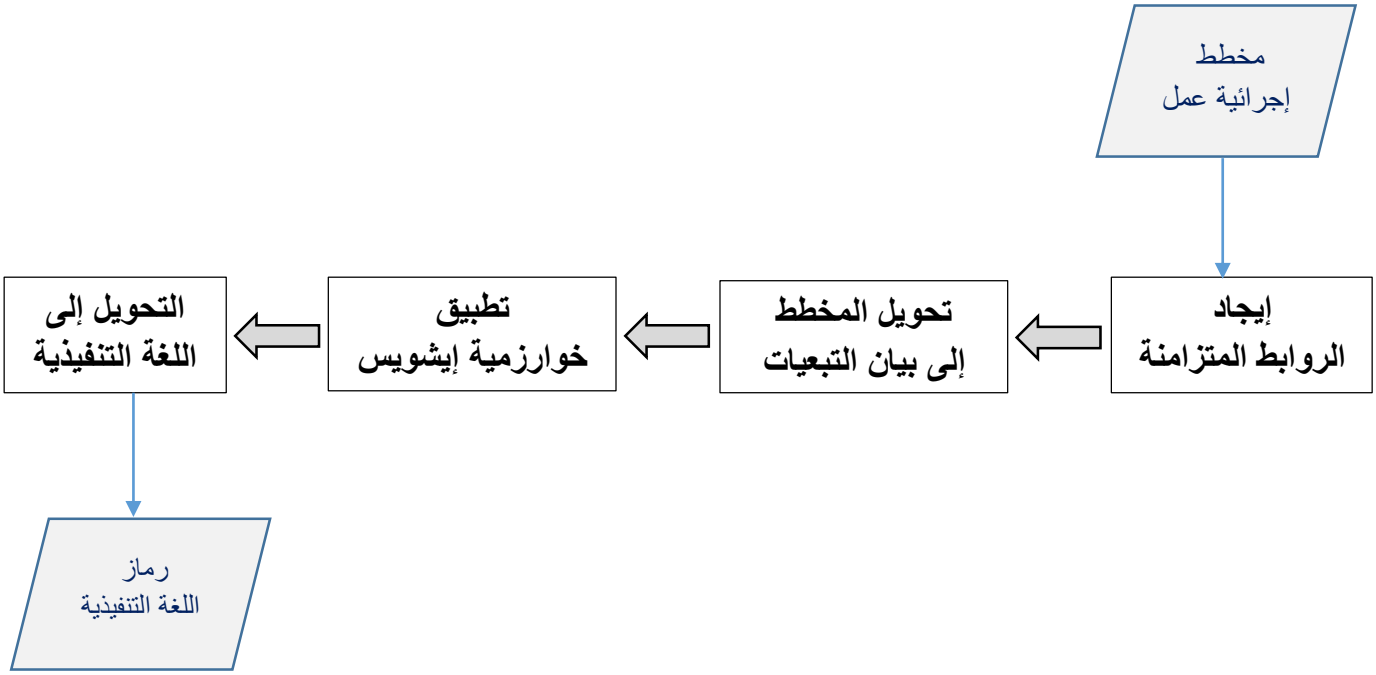
3.1. مقدمة

ناقشنا في القسم السابق الاختلافات بين المنهجيات المتوفرة للانتقال من التمثيل الرمزي إلى اللغة التنفيذية لإجرائيات العمل، فإجرائيات العمل المهيكلة وغير المهيكلة يجري تحويلها إلى لغة تنفيذية وفقا لعدة منهجيات. وقد وجدنا أن المقارنة بين المنهجيات المختلفة تتم من خلال معيارين أساسيين هما **الكمال** (القدرة على تحويل كافة نماذج التمثيل الرمزي إلى اللغة التنفيذية) و**قابلية قراءة** رماز اللغة التنفيذية الناتج، ووجدنا أيضا أن المعيارين السابقين متعارضين أي أن الخوارزميات التي تسعى إلى الكمال تنتج رماز لغة تنفيذية صعب القراءة، بينما الخوارزميات التي يكون هدفها انتاج رماز لغة تنفيذية مقروء فإنها تستطيع تحويل فقط نماذج التمثيل الرمزي المهيكلة. ولأن التوجه العام لمحللي ومصممي النظم لاتباع طرق منهجية والالتزام بالمعايير وبناء التصاميم المضبوطة والمهيكلة، وباعتبار أننا نستطيع فرض بعض الشروط على محليي النظم الذين سيقومون بنمذجة إجرائيات العمل وفق التمثيل الرمزي، فإننا سنولي **اهتمام أكبر لمعيار قابلية القراءة** وذلك نظرا لأهمية هذا المعيار. فرماز اللغة التنفيذية لإجرائيات العمل الناتج عن هذه التحول قد يحتاج أحياناً إلى تنقية وتهذيب وخاصة أثناء عملية الاختبار والتنفيذ. فلو كان القصد من اللغة التنفيذية هو التخاطب مع الآلة فقط وليس للاستخدام البشري، فكان الأفضل التحويل المباشر إلى إحدى اللغات البرمجية الرئيسية المعروفة (C++,C#, Java ....) أو لغة الآلة مباشرة. أي أن صعوبة قراءة رماز اللغة التنفيذية الناتج عن عملية التحويل من شأنه إضعاف لغة BPEL باعتبارها لغة توصيفية لتجميع خدمات الويب [6].

3.2. نموذج الحل

ضمن الدراسة المرجعية السابقة وجدنا أن خوارزمية إيشويس-كرين [16] تقوم بتركيب الخدمات ضمن إجرائيات مهيكلة، أي أنها تساعد بالحصول على بنى مهيكلة لإجرائيات العمل والتي يمكن تحويلها إلى لغة تنفيذية مقروءة بشكل جيد، إلا أنّ دخل هذه الخوارزمية هو بيان تبعيات وليس BPMN. لذلك ينبغي علينا في البداية تحويل مخطط إجرائية العمل إلى بيان تبعيات يشكل دخل لخوارزمية إيشويس. ولأنَّ خوارزمية إيشويس

كغيرها من خوارزميات تحويل البنى غير المهيكلة إلى بنى مهيكلة تعاني من وجود روابط متزامنة بين الفروع المتوازية [20] [21] [22]، لذلك قمنا بإضافة خوارزمية استكشاف لهذه الروابط المتزامنة والتي يمكن تحويلها إلى روابط تحكم ضمن اللغة التنفيذية، أما باقي أجزاء الإجرائية فسيتم معالجتها عن طريق خوارزمية إيشويس والحصول على شجرة إجرائية عمل مهيكلة، لنقوم بعدها بتحويل هذه الشجرة إلى رماز لغة تنفيذية باستخدام خوارزمية مطابقة البنية[5]. والشكل (3.1) يمثل مخطط عمل لنموذج التحويل الذي قمنا ببنائه والذي دخله BPMN وخرجه BPEL.



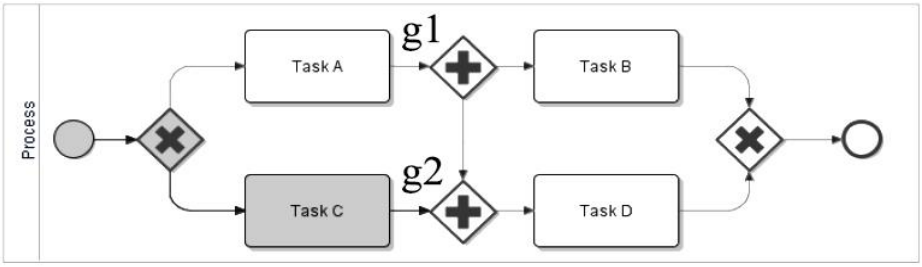
الشكل (3.1) نموذج الحل المقترح لعملية التحويل بين التمثيل الرمزي واللغة التنفيذية

سنبدأ العمل باستخدام مخطط إجرائية العمل المعرف بشكل جيد، وسنناقش شروط البناء الجيد لمخططات إجرائية العمل المستخدمة كدخل للخوارزمية وانعكاس التراخي بهذه الشروط على نتائج خرج الخوارزمية وعن إمكانية قبول الخوارزمية لمخططات المبنية بشكل مقبول وغير متين وكيفية تحويلها إلى رماز اللغة التنفيذية.

3.2.1. تبعيات التزامن

كما ذكرنا فإننا نهدف إلى بناء خوارزمية تدعم الروابط المتزامنة. التزامن الذي سندرسه هو وجود مهمة لا يمكن أن تنفذ إلا بعد انتهاء مهمة أخرى موجودة بفرع آخر من وضـــع التزامن، أما الحالات الأخرى من التزامن فلن نوليها الاهتمام لأنها قد تؤدي في أغلب الأحيان إلى حدوث القفل المميت؛ فمثلا انتظار انتهاء تنفيذ مهمة في فرع لن ينفذ (ناتج عن xor مثلا) سيؤدي بالضرورة إلى حصول القفل المميت. ففي الشكل (3.2) نلاحظ مخططاً لن يكتمل تنفيذه أبداً، فلو فرضنا أن المسار تم عبر المهمة C، فإننا سنصل إلى البوابة g2 التي ستنتظر التفرع الخارج من البوابة g الذي لن يتم باعتبار ان المهمة A لن تنفذ، وبالتالي لن يتم اســـتكمال تنفيذ الإجرائية. بالطبع لا معنى لتحويل نموذج إجرائية العمل الذي يحتوي على القفل المميت (deadlock)، وذلك لأنه سيؤدي إلى رماز لغة تنفيذية غير صالح.

طبعا لن نولي الاهتمام لهذه الروابط لأنها روابط ناتجة عن خطأ من قبل مصمم إجرائية العمل وليست من صلب العمل، أي أنها لا تخدم إجرائية العمل لنقوم بمعالجتها وتحويلها إلى رماز اللغة التنفيذية ويمكن اســـتكشـــافها وحذفها مسبقاً عن طريق شبكات بيتري.



الشكل (3.2) روابط متزامنة خاطئة تؤدي إلى قفل الموت

يوجد أيضاً بعض الشروط التي يفرضها توصيف اللغة التنفيذية [3] وذلك باعتبار أنّه يجري تحويل الروابط المتزامنة السابقة إلى مكونات الرابط <link> وهذه الشروط هي:

1. لا يجوز للروابط المتزامنة أن تتسبب بسلوك حلقي: لأنه ينتج عن ذلك أن منبع تزامن يملك نشاط هدف وبنفس الوقت هو نشاط سابق.
2. لا يجب أن تتجاوز تبعيات التزامن حدود البنى التكرارية.
3. يجب ان يكون هناك بوابة تفريق متوازية قبل كل من منبع وهدف التبعية المتزامنة.

كما أنّ هناك قضية مهمة يجب مراعاتها في تبعية التزامن وهي حذف مسار الموت Dead-Path-Elimination (DPE)، فضمن اللغة التنفيذية يتوجب على مكونات منبع الرابط <link> أن تأخذ قيمة قبل أن تتم معالجة مكونات الهدف. باستخدام حذف مسار الموت يتم إهمال الأنشطة التي لا يتم تنفيذها بوضع قيمتها false، وعندها يتم تنفيذ إجرائية العمل ككل ويتم إنهائها عند الضرورة.

يمكن تحديد الرابط المتزامن المرشح (الممكن تحقيقه للشروط) بسهولة إذا أنه يجب أن يحقق علاقة مباشرة بين بوابة التفريق المتوازي وبوابة التجميع المتوازي. بينما يجري إهمال بقية النماذج للروابط المتزامنة إذا أنها لا تحقق شروط البناء الجيد لمخطط إجرائية العمل التي تحدثنا عنها سابقاً. حيث أنه من غير المسموح استخدام روابط دخل أو خرج متعددة للعقدة الواحدة ضمن مخططات إجرائية العمل المبنية بشكل جيد.

سنقوم بالبداية بالتحقق من أن الرابط الداخل هو رابط متزامن أو لا، حيث سيتم ضمن الخوارزمية المقترحة التعامل مع الرابط المتزامن بطريقة مختلفة عن الروابط من خلال اضافة تبعيات التزامن في التركيب المهيكل الناتج عن تطبيق خوارزمية إيشويس.

3.2.1.1. إيجاد التبعيات المتزامنة

تنتج التبعيات المتزامنة عن وجود روابط مباشرة بين بوابات التفريق والتجميع المتوازية، ففي حال عدم وجود أي علاقة مباشرة بين اثنين من هذه البوابات فلا يوجد أي تبعيات متزامنة [20].

يتم تجميع المهام والأحداث التي تسبق البوابة وكذلك المهام والأحداث التي تلي البوابة وذلك لبوابات التفريق والتجميع المتوازنة.

فالمجموعة before(g)-set تتضمن المهام والأحداث التي يجري تنفيذها قبل الدخول إلى البوابة g.

والمجموعة after(g)-set تتضمن المهام والأحداث التي يجري تنفيذها بعد الخروج من البوابة g.

أما اجتماع المجموعتين السابقتين set branch(g) وهو يمثل جميع المهام والأحداث منذ عقدة الدخول وحتى عقدة الخرج.

سيتم تطبيق الخوارزمية المقترحة على مناطق ذات نقطة دخل واحدة ونقطة خرج واحدة (SESE) والتي سبق تعريفها ضمن القسم الثاني من هذا البحث. ومن أجل كل بوابة تفريق متوازية $f \in G^F$، نبدأ من البوابة

ذات المجموعة before(f)-set الأصـــغر، ونقوم بمطابقتها مع بوابة التجميع المتوازية $j \in G^J$، حيث يجري التطابق عندما تحتوي المجموعتين نفس العناصـــر branch(f) = branch(j)، إذا لم يكن هناك أي بوابة تجميع تحتوي نفس عناصر بوابة التفريق، فإن واحدة من المجموعتين تكون مجموعة جزئية في المجموعة الثانية branch(f) ⊆ branch(j) OR branch(f) ⊇ branch(j).

فالرابط بين بوابات التفريق والتجميع المتطابقة ليس رابطاً متزامناً ويتم حذف بوابة التجميع من المجموعة التي تحدد روابط التزامن، وهذا موضح ضمن الخوارزمية (3.5) والتي تقوم أيضا بترتيب مجموعة البوابات باستخدام الحلقة while، حيث تؤخذ البوابات الأصـــغر بالنســـبة لبوابات التفريق والبوابات الاكبر من أجل بوابات التجميع، ويتم انهاء حلقة while عندما يتم معالجة كل البوابات. يســاعد وجود الترتيب في تنفيذ الخوارزمية بفعالية أكبر لأن مطابقة البوابات تم ســـابقا. فبوابة التجميع j التي لا يمكن مطابقتها مع أي بوابة تفريق f وحيث توجد علاقة مباشـــرة بين بوابة التفريق f وبوابة التجميع j يتم تحديدها على أنها هدف للرابط متزامن. ويكون مصدر الرابط المتزامن هو بوابة التفريق f للعلاقة المباشرة.

يتم تخزين كافة الروابط المتزامنة ضمن المجموعة $F_S \subseteq F$ ليتم معالجتها بشكل مختلف عن العلاقات الأخرى في المرحلة اللاحقة.

في البداية ســـنقوم بشـــرح خوارزمية إيجاد الروابط المتزامنة ومن ثم ســـنقوم بتطبيق هذه الخوارزمية على مخطط معالجة الشكوى الذي سبق شرحه.

خوارزمية إيجاد الروابط المتزامنة الشـــكل (3.3)، تأخذ كدخل مجموعة من بوابات **التفريق** المتوازية G^F، والتي يتم ترتيبها **تصاعديا** وفق لحجم المجموعة before(g)-set، ومجموعة من بوابات **التجميع** المتوازية G^J، والتي يتم ترتيبها **تنازلياً** وفق لحجم المجموعة before(g)-set كما ذكرنا ســابقاً. حيث تقوم الخوارزمية في البداية بمطابقة branch(g)-sets وفي حالة عدم وجود تطابق فإنه تتم عملية المطابقة للمجموعات الجزئية من بوابات التفريق والتجميع للمجموعات branch(g)-sets.

في حالة عدم وجود تطابق بين بوابات التجميع J وبوابات التفريق F، فإنه يتم تحديد الرابط المتوازي عندما يوجد علاقة مباشرة بين البوابتين في F.

طبعا لدينا حلين بديلين عن تطبيق خوارزمية استكشاف التبعيات المتزامنة، فيمكن سؤال المستخدم ليقوم بتحديدها مسبقا، أو يمكن معالجة جميع المكونات التي تحتوي تبعيات متزامنة كمكونات غير مهيكلة سواء أكانت هذه المكونات مهيكلة أو غير مهيكلة.

تخلينا عن الخيار الأول لأننا نريد بناء نموذج يتطلب أقل قدر ممكن من تدخل المستخدم، وأما الخيار الثاني الذي يقوم على مبدأ معالجة كافة المكونات على أساس أنها غير مهيكلة كما في [15]، فإن هذا ينقص كثيرا من قابلية القراءة لرماز اللغة التنفيذية الناتج حيث تم تحويل كل العلاقات ضمن هذا النموذج إلى روابط.

```
Find_Synchronize (G^F,G^J,F)
links := ∅
if ∃(f, j) ∈ F | f ∈ G^F ∧ j ∈ G^J then
    forktovisit := G^F
    while forktovisit ≠ ∅ do
        f := the gateway with the smallest before(g)-set of forktovisit
        SearchSubsets = true
        jointovisit := G^J
        while jointovisit ≠ ∅ do
            j := the gateway with the largest before(g)-set of jointovisit
            if branch(f) = branch(j) then
                G^J := G^J\{j}
                SearchSubsets = false
                break;
            end if
            jointovisit := jointovisit\j
        end while
        if SearchSubsets = true then
            while jointovisit ≠ ∅ do
                j := the gateway with the largest before(g)-set of jointovisit
                if branch(f) ⊇ branch(j) ∧(f, j) ∉ F then
                    G^J := G^J\{j}
                    break;
```

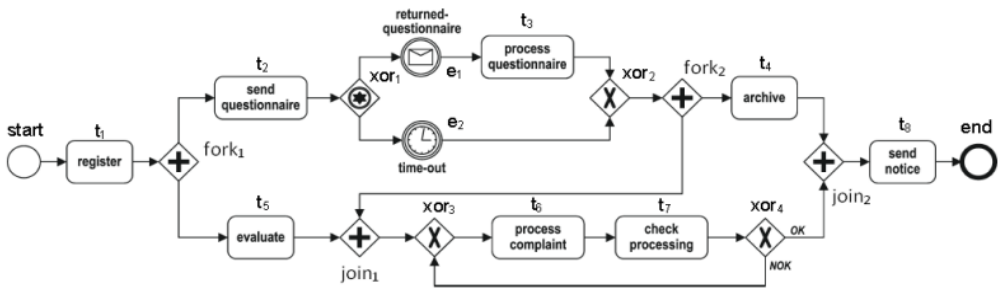
```
                    else if branch(f) ⊆ branch(j) ∧(f, j) ∉ F then
                            G^J := G^J\{j}
                            break;
                    else if (f, j) ∈ F then
                            links := links ∪ {(f, j)}
                    end if
                    jointovisit := jointovisit\j
                end while
            end if
            forktovisit := forktovisit\f
        end while
end if
return links
```

الشكل (3.3) خوارزمية إيجاد الروابط المتزامنة

طالما أننا سنتعامل مع الروابط المتزامنة بشكل مختلف عن الروابط الأخرى، فإنه سيلزمنا تعريف **مخطط إجرائية العمل المعدل** وهو مخطط إجرائية العمل الخالي من الروابط المتزامنة والتي سيتم الاستعاضة عن البوابات فيه بعقد وهمية. فلتكن F^N مجموعة العلاقات باستثناء العناصر ضمن الروابط المتزامنة F^S أي أي $F^N = F\backslash F^S$. يمكن أن تتحول بعض البوابات إلى بوابات ذات مدخل واحد ومخرج واحد، والذي يتعارض مع التعريف الجيد لمخطط إجرائية العمل. وباستبدال هذه البوابات بمهام وهمية، نحصل في النهاية على مخطط إجرائية عمل معرف بشكل جيد.

3.2.1.2. مثال: إجرائية معالجة الشكوى

سنستخدم لشرح الخوارزمية مخطط إجرائية معالجة الشكوى الشكل (3.4) التي سبق شرحه وهذه الإجرائية تحتوي على تزامن، أحداث، روابط متزامنة، بنى تكرارية، ...إلخ. إي أن هذه الإجرائية تحتوي على مختلف بنى BPMN لكنها لا تتجاوز شروط البناء الجيد لمخطط إجرائية العمل.



الشكل (3.4) إجرائية معالجة الشكوى

ففي البداية سنقوم بحساب المجموعتين before(g)-set, after(g)-set لكل من بوابات التفريق والتجميع المتوازية وهذه النتيجة موضحة في الجدول (3.1)، حيث يتم ترتيب مجموعة بوابات التفريق المتوازية G^F بشكل تصاعدي بالنسبة before(g)-set فنحصل على $G^F = \{fork_1, fork_2\}$ ويتم ترتيب مجموعة بوابات التجميع المتوازية G^J بشكل تنازلي بالنسبة ل before(g)-set فنحصل على $G^J = \{join_2, join_1\}$

البوابة	before(g)-set	after(g)-set
fork1	----	t2, e1, t3, e2, t4, t5, t6, t7
Fork2	t2, e1, t3, e2	t4, t6, t7
join1	t2, e1, t3, e2, t5	t6, t7
join2	t2, e1, t3, e2, t4, t5, t6, t7	---

الجدول[6] (3.1) المجموعتين before(g)-set, after(g)-set لبوابات مخطط إجرائية العمل

بدراسة الجدول السابق نستنتج إنه يوجد تطابق بالمجموعة branch(g)-set بين بوابة التفريق fork1 وبوابة التجميع join2، وبالتالي لا يمكن أن يكون هناك تبعية تزامن بين البوابتين.

أما بالنسبة للبوابتين join1, fork2 فنلاحظ عدم وجود تطابق بالمجموعة branch(g)-set بينهما، وأيضاً branch(fork2) لا تشكل مجموعة جزئية من branch(join1) وكذلك branch(join1) لا تشكل

[6] تم حساب المجموعتين before(g)-set, after(g)-set ضمن منطقة (SESE)، لذلك لم تظهر الاحداث والمهام (start, t1 ,t8, end) ضمن الجدول (3.1)

مجموعة جزئية من branch(fork2). وبما أن العلاقة بين fork2, join1 هي علاقة مباشرة $\exists(f, j) \in F \mid f \in G^F \wedge j \in G^J$ وبالتالي فإنه يتم تحديد هذه العلاقة بأنها رابط متزامن.

3.2.2. العقدة المهيمنة، رأس الحلقة، المجموعات اللاحقة

لإيجاد هيكلية مخطط إجرائية العمل، يتم تعريف بعض المفاهيم [16] كالعقدة المهيمنة (Dominators) رأس الحلقة (loop headers)، المجموعات اللاحقة (follow sets)، حيث يمكن حساب المجموعات اللاحقة وتكون النتيجة هي توصيف للخصائص الهيكلية لمخطط إجرائية العمل. وسيتم استخدام هذه المفاهيم لاحقا بتوصيف الخوارزمية المقترحة.

يستخدم مفهوم المهيمن لتحديد البنية المتداخلة[16]، ولشرحها نفرض لدينا العقدتين $p, q \in O$ من الأغراض O. والبداية هي عقدة من E^S، نقول إن العقدة p تهيمن على العقدة q في حال كان كل مسار من البداية إلى العقدة q يمر من p، وهذا المسار خال من الروابط المتزامنة.

كل عقدة ماعدا عقدة البداية لديها على الأقل عقدة مهيمنة واحدة. العقدة p هي عقدة مهيمنة مباشرة للعقدة q إذا كان كل مهيمن للعقدة q غير p هي عقدة مهيمنة أيضا ل p أيضاً. كل عقدة تملك عقدة مهيمنة مباشرة واحدة [19]، وسنرمز لهذه العقدة بالرمز DOM(p).

أما بالنسبة للحلقات فيتم تحديدها بمساعدة الوصلات الرجعية، فالوصلات الرجعية edge (x, y) في F^N حيث y تهيمن على x. الحلقات الطبيعية من الوصلات الرجعية يتم حسابها بسهولة وهي مجموعة العقد التي تصل x بدون المرور من y، بالإضافة ل y. العقدة y هي رأس للحلقة الطبيعية ويرمز لها HEAD(n) = y. العقدة الهدف لبعض الوصلات الرجعية تسمى عقدة الحلقة. في حال كانت العقدة ليست عقدة حلقة، فإن رأس الحلقة HEAD(n) غير معرف.

عندما نقوم بتحديد العقد المهيمنة ورؤوس الحلقة لمخطط إجرائية العمل، فإننا نستطيع حساب المجموعات اللاحقة follow-sets. فالمجموعات اللاحقة ل p تتضمن جميع العقد التي تلي p مباشرة بنفس مستوى التداخل في التركيب المهيكل.

Definition 3. The follow-set of p, **FOLLOW(P)**, can be calculated with the following rules**[16]**:

For a fork node p, so $p \in G^F \cup G^D \cup G^V$, let

$$FOLLOW(p) = \{ q \mid q \in G^J \cup G^M \wedge p = DOM(q) \wedge HEAD(p) = HEAD(q) \}$$

For a loop node p, so some back edge enters p, define

$$FOLLOW(p) = \{ q \mid DOM(q) \text{ is in a natural loop headed by } p \}$$

For each other node p, define

$$FOLLOW(p) = \{ q \mid HEAD(p) = HEAD(q) \wedge p = DOM(q) \}$$

3.2.3. قواعد اللغة التنفيذية BPEL

بعد تحديد كل من الروابط المتزامنة والعقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة، نكون بذلك قد حصلنا على المعلومات الضرورية التي تشكل دخل الخوارزمية، وسيكون خرج الخوارزمية توصيف إجرائيات اللغة التنفيذية بقواعد اللغة التي سيتم ذكرها الأن.

للحفاظ على خرج الخوارزمية قابل للقراءة والفهم، يتوجب علينا في الخرج التقيد بالنحو المحدد ضمن [16]، حيث سنقوم بعملية دمج بين هذا النحو ونحو اللغة التنفيذية المباشر لنحصل على خرج رماز لغة تنفيذية لإجرائية العمل قابل للقراءة ومفهوم.

لتكن t مهمة من مجموعة المهام T، و e حدث من مجموعة الاحداث E. ولتكن d غرض من مجموعة الأغراض الوهمية D. فتكون لغة الإجرائية المهيكلة P معرفة بالنحو التالي

P ::= seq | and{seq, seq, .., seq, linkset} | xor{grd▷ seq, grd▷ seq, .., grd▷ seq}
| repeat seq until grd | atomic

atomic ::= t | e | d

seq ::= < P, P, .., P >

grd ::= in(atomic) | grd ∨ grd | true

linkset ::= {link(atomic, atomic), ..,link(atomic, atomic)}

- التعبير < P, P, .., P > يعني أنه يتم تنفيذ العناصر ضمن القائمة بالترتيب المذكور ضمن القائمة.
- التعبير and{seq, seq, .., seq, linkset} يحدد أن العناصر ضمن القائمة يتم تنفيذها بالتوازي والفروع يمكن أن تملك عدة تبعيات متزامنة والتي يجري تعريفها ضمن linkset.
- التعبير xor{grd▷seq, grd▷seq, .., grd▷seq} يحدد أنه هناك واحد فقط من التعابير المحددة ضمن المجموعة يتم تنفيذه.
- التعبير repeat seq until grd يعني أنه يتم تكرار تنفيذ التعبير seq حتى يتم تحقيق الشرط grd.
- الشرط in(atomic) يكون محقق true عندما يكون قد تم تنفيذ atomic مسبقا.

فالتابع grd يستخدم لتمثيل تعابير guard والتي تعني أن خدمة لا يتم تنفذيها إلا إذا تم تنفيذ الخدمة السابقة لها مباشرة.

For a service x, **guard in(x)** is true if x was executed previously.

- المجموعة linkset يمكن أن تحوي العديد من التبعيات المتزامنة، والتي يتم تعريفها بالشكل link(atomic, atomic) والتي تعني أن تبعية التزامن بين العنصرين atomic يكون الأول مصدر التزامن والثاني هدف، فالعنصر الهدف لا يمكن معالجته إلا بعد تفعيل الرابط من قبل العنصر المصدر. يجب أن يكون كلا العنصرين ضمن نفس المجال (scope)، وهو الشكل المختصر للبنية and.

3.2.4. الخوارزمية

سنقوم في البداية بشرح طريقة تعديل خوارزمية إيشويس ومن ثم سنقوم تطبيق الخوارزمية المعدلة على مخطط إجرائية معالجة الشكوى المذكور سابقاً

3.2.4.1. تعديل خوارزمية إيشويس

سنقوم بتعديل خوارزمية إيشويس [16] لتكون معاملات الدخل مناسبة لمخطط إجرائية العمل المبني بشكل جيد ولتسمح بإضافة الروابط المتزامنة.

معاملات الدخل:

i. مجموعة من الأغراض O.
ii. مجموعة من العلاقات F^N.
iii. مجموعة من الروابط المتزامنة F^S.
iv. تابع جزئي grd يحول العلاقات إلى تعابير guard.
v. العقدة الحالية current التي سيتم معالجتها.

وكما ذكرنا سابقا فإن الخوارزمية تحتاج إلى مخطط إجرائية عمل معرف بشكل جيد كدخل، ومن أجل كل عقدة تفريق، نحتاج إلى عقدة تجميع تابعة من نفس النوع. ويمكن أن تكون عدة عقد تجميع تابعة لعقدة تفريق واحدة أو العكس أي عقدة تجميع واحدة تخص مجموعة من عقد التفريق لكن بشرط أن تكون من نفس النوع [16].

كل الحلقات تحتاج لنقطة نقطة دخل واحدة $g^{entry} \in G^M$ ونقطة خرج $g^{exit} \in G^D \cup G^V$ واحدة، كما يتوجب على التزامن ألا يتجاوز حدود الحلقة لأن هذا السلوك غير مقبول ضمن الإجرائيات المهيكلة.

يجب تطبيق خوارزمية إيجاد التبعيات المتزامنة (الشكل 3.6) قبل خوارزمية إيشويس، ويتألف خرج خوارزمية إيشويس [16] من مجموعة من الكتل التسلسلية (sequential block) والتي تبدأ بالعقدة الحالية (current) وترتكز على قواعد اللغة التنفيذية التي سبق شرحها.

```
procedure StructuredComposition (O, F^N, F^S, grd , current) [16]
    if current ∈ G^F ∪ G^D ∪ G^V then
        children := ∅
        for n ∈ post(current) do
            if n not in any FOLLOW set then
                C_n := StructuredComposition (O, F^N, F^S, grd, n)
            Else
                C_n :=< dummy_{current,n} >
            end if
```

```
            if current ∈ G^D ∪ G^V then
                C_n := grd(current, n) ▷ C_n
            end if
            children := children ∪ {C_n}
        end for
        if current ∈ G^D ∪ G^V then
            P_comp := xor children
        else
            C_n := AddingSynchronization(children, F^S)
            children := children ∪ {C_n}
            P_comp := and children
        end if
        P := < P_comp >
    else if current is loop node with successor node x and FOLLOW node n then
        P_x := StructuredComposition(O, F^N, F^S, grd, x)
        P :=< repeat P_x until in(dummy_current,n) >
    else
        P :=< current >
    end if
    if FOLLOW(current) ≠ ∅ then
        if |FOLLOW(current)| > 1 then
            insert unique FOLLOW node after current
        end if
        next := the unique node following current
        Q := StructuredComposition(O, F^N, F^S, grd, next)
        P := P⌢Q
    end if
    return P
end procedure
```

الشكل (3.5) خوارزمية بناء المكونات المهيكلة

والشكل 3.6 يوضح خوارزمية إضافة الروابط المتزامنة والمبنية على أساس الخوارزمية [16]، ودخلها مجموعة من الأغراض من البنى (children) and-construct ومجموعة من الروابط المتزامنة والتي لم يتم ذكرها في توصيف الإجرائية. إذا تم ذكر كافة الروابط المتزامنة ضمن توصيف الإجرائية فسيتم تجاوز الخوارزمية باعتبار مجموعة الروابط المتزامنة المتبقية هي مجموعة خالية، وإلا فإنه سيتم تحليل مجموعة الأبناء (children) وسيتم إيجاد بوابات التجميع والتفريق. إذا وجدت كلا النوعين من البوابات ضمن مجموعة الأبناء فإنه سيتم وضع رابط ضمن المجال.

```
procedure AddingSynchronization ((children,F^S))
    if (F^S)) = ∅) then
        return ∅
    else
        linkset := ∅
    for ( (f, j) ∈ F^S ) do
        if f ∈ children ∧ j ∈ children then
            F^S := F^S \(f, j)
            linkset := linkset ∪ {link(f, j)}
        end if
    end for
    return linkset
    end if
end procedure
```

الشكل (3.6) خوارزمية إضافة الروابط المتزامنة

procedure **Inserting unique FOLLOW node after current [16]**

f := a fresh node not in O
O := O ∪ {f}
$E_{follows}$:= { (x, y) | (x, y) ∈ F^N ∧ x ∈ O ∧ y ∈ FOLLOW(current)}
E_{new} := { (x, f), (f, y) | (x, y) ∈ $E_{follows}$ }
E := (F^N \ $E_{follows}$) ∪ E_{new}
if current ∈ G^D ∪ G^V then
grd := grd ∪ {(f, y) → formulaPreCondition(y) | (x, y) ∈ $E_{follows}$ }
end if

end procedure

الشكل (3.7) خوارزمية إدخال عقدة تابعة وحيدة بعد العقدة الحالية [16]

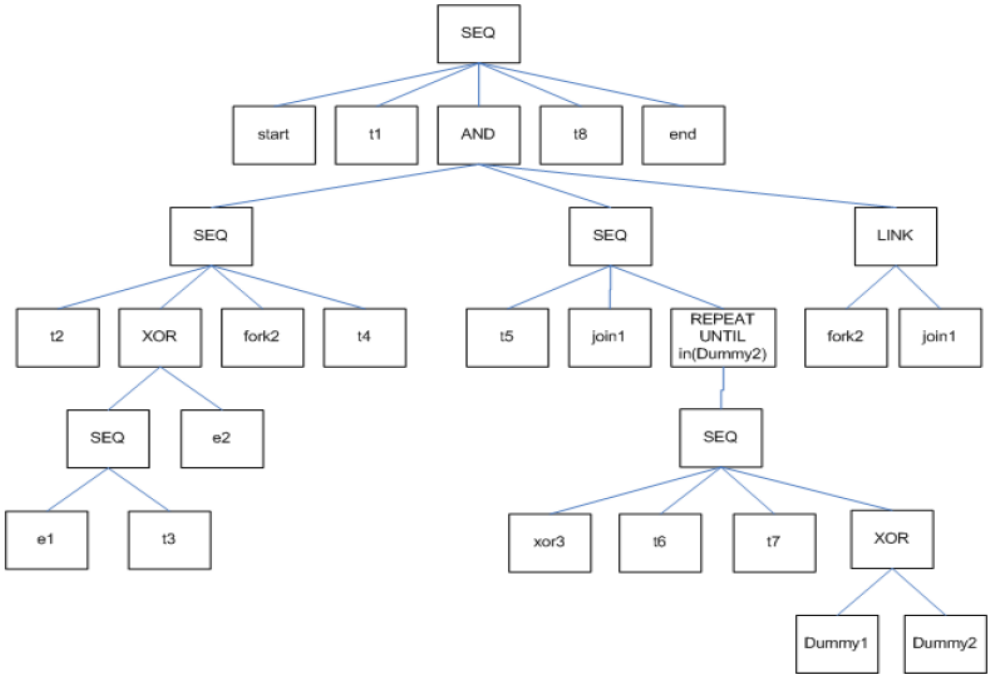
3.2.4.2. تطبيق الخوارزمية على مخطط إجرائية معالجة الشكوى

لتوضيح الخوارزمية السابقة، سنقوم بتطبيقها على مخطط إجرائية العمل الخاص بمعالجة الشكوى الموضح ضمن الشكل (3.4)، قد قمنا سابقا بإيجاد العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة في الجدول (3.1). سنستخدم مجموعة العلاقات F^N بدلا من F باعتبار أننا لن نقوم بمعالجة الروابط المتزامنة الأن، حيث نلاحظ أن كل عقدة أخيرة من مستوى التداخل تملك مجموعة خالية FOLLOW(n)-set، بينما باقي العقد الأخرى فإنها تملك عنصر وحيد ضمن المجموع FOLLOW(n)-set. الحلقة توجد بين العقدتين xor3,xor4.

وفقا لهذه المفاهيم فإنه يمكن تنفيذ الخوارزمية والحصول على التراكيب المهيكلة، والشكل (3.8) يوضح التراكيب المهيكلة الناتجة عن تطبيق الخوارزمية، بالأجمال فإنه تم إضافة عقدتين وهميتين لدعم الحلقة في النموذج المهيكل، وقد تم قبلها إضافة رابط متزامن إلى التركيب المهيكل، والرابط المتزامن كما هو واضح ضمن الشكل (3.8) مصدره fork2 وهدفه join2 ضمن مجال البنية and-construct.

العقدة	العقدة المهيمنة	رأس الحلقة	المجموعات اللاحقة
start	-	-	t1
t1	start	-	fork1
fork1	t1	-	join2
t2	fork1	-	xor1
xor1	t2	-	xor2
e1	xor1	-	t3
t3	e1	-	-
e2	xor1	-	-
xor2	xor1	-	fork2
fork2	xor2	-	t4
t4	fork2	-	
t5	fork1	-	join1
join1	t5	-	xor3
xor3	join1	-	t6
t6	xor3	xor3	t7
t7	t6	xor3	xor4
xor4	t7	xor3	-
join2	fork1	-	t8
t8	join2	-	end
end	t8	-	-

الجدول (3.2) العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة



الشكل (3.8) التراكيب المهيكلة الناتجة لمخطط إجرائية معالجة الشكوى

3.2.5. نحو اللغة التنفيذية لإجرائية العمل BPEL syntax

إن التركيب المهيكل الناتج يمثل إجرائية العمل بالتمثيل BPEL، إلا أن ترميزه يختلف عن الترميز الموضح سابقا، وبالتالي فإننا نحتاج إلى مرحلة أخيرة للتحويل إلى الترميز المنشود للغة التنفيذية.

يمكن تحويل كل عنصر من عناصر التركيب المهيكل إلى رماز اللغة تنفيذية، فمثلا العقدة (SEQ) تتحول إلى العنصر <sequence>، والعقدة (AND) تتحول إلى العنصر <flow>. والعمود الأخير من الجدول يحتوي على بعض المعلومات الإضافية اللازمة من أجل التصريح بأن رماز لغة تنفيذية صحيح (valid). وهذه المعلومات تحتاج إلى تدخل يدوي من المستخدم ولا يمكن الحصول عليها من نماذج BPMN.

نلاحظ ضمن الجدول وجود احتمالين لتحويل بنية XOR، وإذا حدث سياق الشرط بين عدة أحداث فإن خيار الحدث يحول إلى البنية <pick>، بينما يحول إلى البنية <if> إذا كان الخيار معتمد على البيانات وليس على الحدث.

Structured composition	BPEL construct	Information needed
task	<invoke>	partnerLink, portType,operation
SEQ	<sequence>	
AND	<flow>	
XOR	<if>,<condition> <elseif>,<condition> <else> <pick>	Condition Condition
LOOP	<repeatUntil>	Condition
LINK	<links>,<link> <sources>, <source> <targets>, <target>	Name linkName

الجدول (3.3) تحويل التركيب المهيكل إلى رماز اللغة التنفيذية

3.3. التنجيز البرمجي لنموذج الحل

يتم تنجيز الخوارزمية ضـــمن بيئة التطوير Eclipse، وقد تم اختيار هذه البيئة لوجود إمكانية لنمذجة إجرائيات العمل باستخدام التمثيل الرمزي BPMN ضـمن Eclipse، بالإضـافة إلى وجود بنى تسـاعد على عملية النمذجة (Graph, Tree).

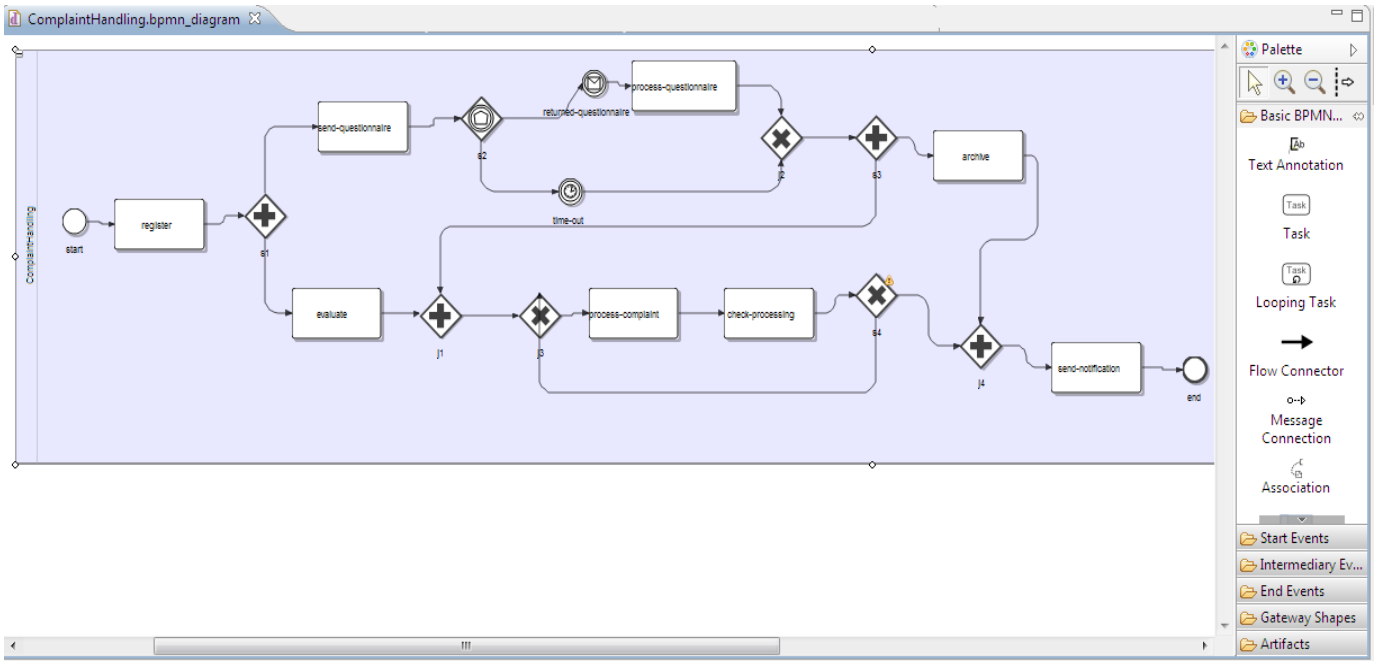
3.3.1. بناء نماذج التمثيل الرمزي لإجرائيات العمل باستخدام الEclipse

قبل البدء بإجراء عملية التحويل بين التمثيل الرمزي واللغة التنفيذية، نحن بحاجة إلى أداة لنمذجة إجرائيات العمل تدعم BPMN، ومن أجل ذلك فإننا نحتاج إلى إضافات Eclipse add-on التالية:

i. Graphical Editors and Frameworks
ii. Models and Model Development
iii. STP BPMN modeler

الإضـافة STP BPMN modeler تدعم فقط النماذج المعيارية للتمثيل الرمزي، ويتم رسـم هذه النماذج عن طريق اختيار بنى جاهزة موجودة ضـمن البيئة وربط هذه البنى ببعضـها عن طريق الأسـهم. والشـكل (3.9) يوضح مثال عن نموذج BPMN مرسوم عن طريق هذه الإضافة ضمن Eclipse.

لكن الإضـافة لا تختبر صـحة نماذج التمثيل الرمزي، فيمكن للمسـتخدم رسـم نماذج خاطئة أو غير مكتملة لتقوم الإضـافة بتحويلها إلى رماز اللغة التنفيذية لإجرائيات العمل، لذلك يتوجب على المسـتخدم أن يكون على دراية بهذه المحدودية ضـــمن الإضـــافة وأن يختبر نماذجه بنفســـه من حيث تحقيقها شـــروط البناء الجيد للنموذج قبل إدخالها للبرنامج من اجل تحويلها إلى رماز اللغة التنفيذية.

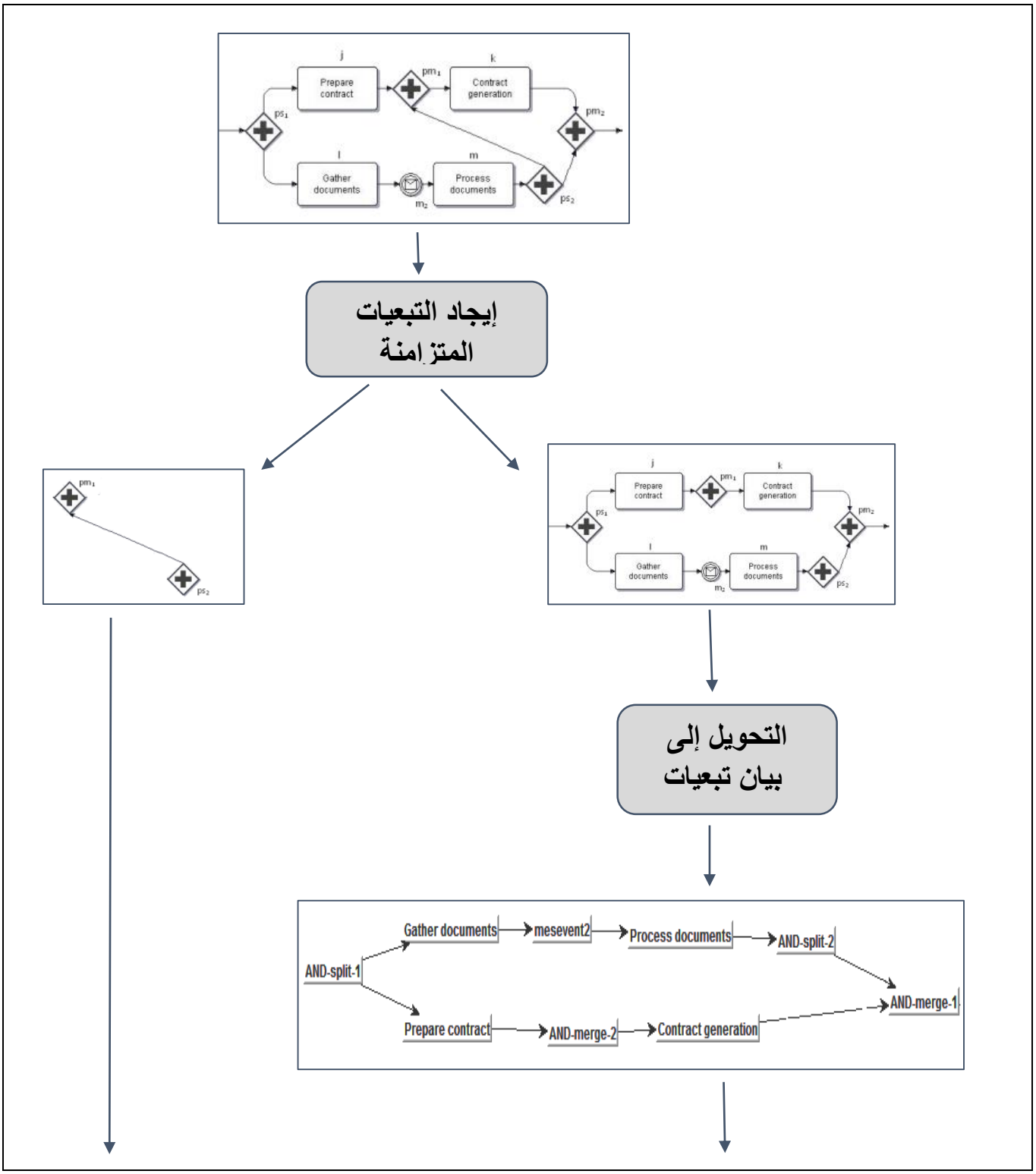


الشكل (3.9) نموذج BPMN مرسوم باستخدام الإضافة ضمن Eclipse

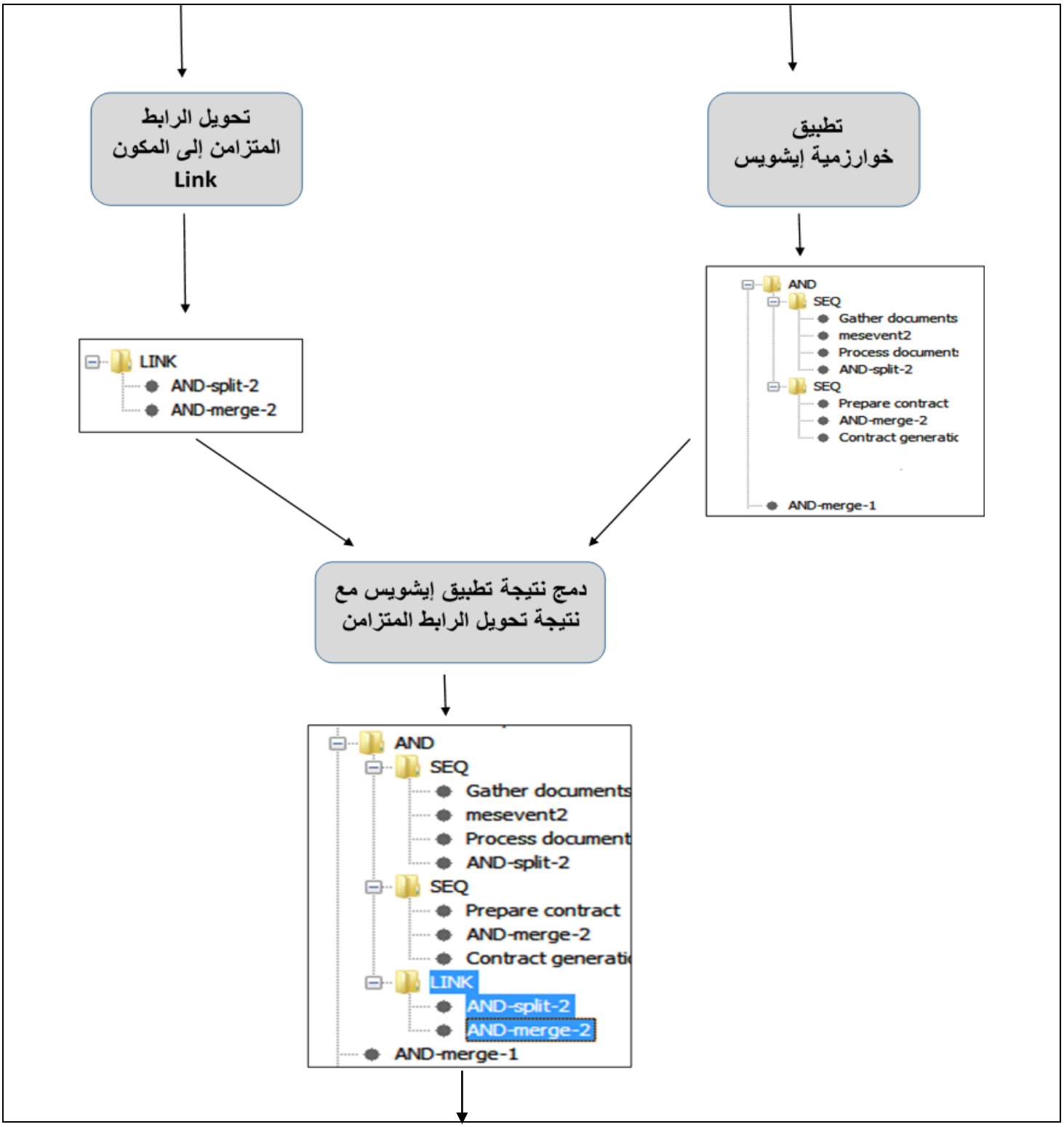
3.3.2. تنجيز الخوارزمية

يتم تنجيز الخوارزمية المطروحة في القسم السابق وفق النموذج الموضح ضمن الشكل (3.10) والتي تمثل تسلسل خطوات التنجيز البرمجي وكما ذكرنا سابقا فإن هذا التنجيز يعتمد على خوارزمية إيشويس [16]. في البداية سيكون الدخل نموذج BPMN، ليتم بعدها تحليل النموذج إلى عقد ووصلات والتي يمكن اعتبارها شكل اخر من بيان التبعية. ولتجنب فقدان بعض المعلومات الموجودة ضمن نموذج التمثيل الرمزي فإن بعض المعلومات يمكن ان يتم اضافتها إلى عقد البيان، فيتم تعريف نوع للعقدة والتي يمكن ان تكون مهمة أو بوابة تفريق على التوازي أو بوابة تجميع.

وبعد ذلك سيتم تطبيق خوارزمية استكشاف التبعيات المتزامنة والتي سيتم تنجيزها على شكل تابع يقوم بالبحث عن هذه التبعيات، وفي حال وجودها يتم استخلاصها وحذفها بشكل مؤقت من النموذج، ليتم استخدامها بعد تطبيق خوارزمية إيشويس وتوليد التركيب المهيكل للإجرائية. فنحصل على شجرة تراكيب مهيكلة للخدمات المشكلة لنموذج إجرائية العمل والعلاقة بين هذه الخدمات. ليتم بعد ذلك تحويل هذه الشجرة إلى رماز لغة تنفيذية قابل للقراءة باعتباره ناتج عن تحويل بنى مهيكلة.



الشكل (3.10) رسم توضيحي لعمل التنجيز البرمجي



الشكل (3.10) متابعة الرسم التوضيحي لعمل التنجيز البرمجي

```
▼<flow>
 ▼<links>
    <link name="AND-split-2-to-AND-merge-2"/>
  </links>
 ▼<sequence>
    <invoke name="Prepare contract" partnerLink="client" portType="tns:PaymentProcessorServiceCallback"
    operation="onInsufficientFundException"/>
   ▼<invoke name="AND-merge-2" partnerLink="client" portType="tns:PaymentProcessorServiceCallback"
    operation="onInsufficientFundException">
     ▼<targets>
        <target linkName="AND-split-2-to-AND-merge-2"/>
      </targets>
    </invoke>
    <invoke name="Contract generation" partnerLink="client" portType="tns:PaymentProcessorServiceCallback"
    operation="onInsufficientFundException"/>
  </sequence>
 ▼<sequence>
    <invoke name="Gather documents" partnerLink="client" portType="tns:PaymentProcessorServiceCallback"
    operation="onInsufficientFundException"/>
    <invoke name="mesevent2" partnerLink="client" portType="tns:PaymentProcessorServiceCallback"
    operation="onInsufficientFundException"/>
    <invoke name="Process documents" partnerLink="client" portType="tns:PaymentProcessorServiceCallback"
    operation="onInsufficientFundException"/>
   ▼<invoke name="AND-split-2" partnerLink="client" portType="tns:PaymentProcessorServiceCallback"
    operation="onInsufficientFundException">
     ▼<sources>
        <source linkName="AND-split-2-to-AND-merge-2"/>
      </sources>
    </invoke>
  </sequence>
</flow>
```

الشكل (3.10) متابعة الرسم التوضيحي لعمل التنجيز البرمجي

الفصل الرابع

الحالات المدروسة

4. الحالات المدروسة

سيتم ضمن هذا القسم مناقشة حالتين، حيث سيتم توصيف الحالتين ومن ثم تحويلهما من النموذج BPMN إلىBPEL، بداية بطريقة يدوية باستخدام الخوارزمية المقدمة في القسم الثالث من هذا البحث ومن ثم باستخدام التنجيز البرمجي.

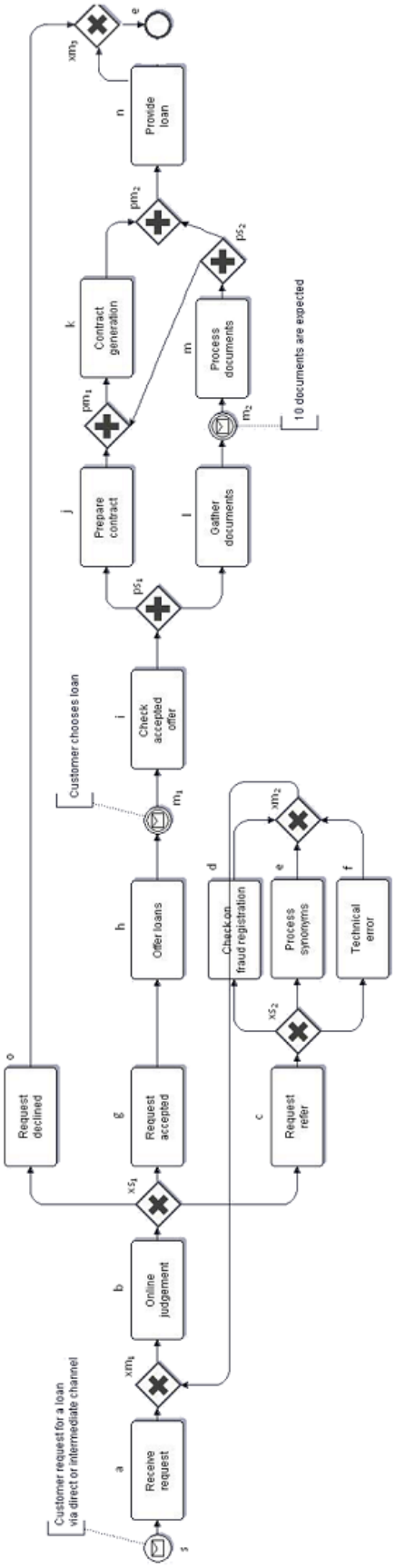
4.1. الحالة المدروسة الأولى: معالجة طلب قرض

الحالة المدروسة الأولى التي سندرسها ستكون معالجة طلب قرض لزبون، فبعد تلقي الطلب وتخزينه ضمن قواعد المعطيات، يتم في البداية معالجة الطلب من خلال واجهة أولية، تقوم بتحديد وضع حالة الطلب ضمن إحدى الحالات الثلاث (الرفض، القبول، إعادة الطلب)، وإعادة الطلب تكون طبقا لأحد الأسباب التالية (شبه تزوير، غموض بالإجرائية، خطأ تقني) وبحسب السبب يتم استدعاء أحد الأنشطة لحل المشكلة أو لطلب معلومات إضافية وبحسب المعلومات الإضافية المستدعاة يتم معالجة الطلب من جديد عن طريق الواجهة السابقة. أما في حالة قبول طلب القرض، فيتم إعلام الزبون بالقروض المتوفرة ليقوم باختيار القرض المناسب له، ليتم بعدها بالتوازي تحضير العقد وجميع الملفات الضرورية، ليتم بعدها توقيع العقد ومنح القرض.

4.1.1. التحويل اليدوي

سنقوم الان بإجراء التحويل اليدوي للتمثيل الرمزي لإجرائية العمل آنفة الذكر والممثلة بالشكل (4.1) إلى تراكيب مهيكلة ممثلا باللغة التنفيذية لإجرائية العمل باستخدام الخوارزمية المذكورة بالقسم السابق.

ففي البداية سنقوم بتطبيق خوارزمية إيجاد التبعيات المتزامنة (الشكل 3.5) ومن ثم سنقوم باستخدام خوارزمية بناء التراكيب المهيكلة (الشكل 3.7)، وسنقوم بمقارنة نتيجة هذا التحويل مع نتيجة التطبيق باستخدام التنجيز المطروح ضمن القسم 3، وكذلك سنقارنها مع الهدف المنشود من هذا التحويل وهو الحصول على رماز لغة تنفيذية قابل للقراءة.



الشكل (4.1) الحالة المدروسة الأولى: طلب قرض

قبل تطبيق خوارزمية إيجاد التبعيات المتزامنة، سنقوم بالتأكد من وجود روابط مباشرة بين بوابات التفريق المتوازية مع بوابات التجميع المتوازية، ففي حال عدم وجود مثل هذه الروابط فليس هناك حاجة لتطبيق الخوارزمية وذلك بسبب شرط وجود رابط مباشر بين بوابة التفريق وبوابة التجميع ليكون هناك رابط متزامن.

بداية سنقوم بتحديد المجموعتين before(g)-set,after(g)-set لبوابات التفريق والتجميع المتوازية والنتيجة في الجدول (4.1)

البوابة	**before(g)-set**	**after(g)-set**
ps1	---	**j, pm1, k, l, m2, m, ps2, pm2**
ps2	ps1, l, m2, m	pm1, k, pm2
pm1	ps1, j, l, m2, m, ps2	k, pm2
pm2	**ps1, j, pm1, k, l, m2, m, ps2**	---

الجدول (4.1) المجموعتين before(g)-set,after(g)-set

لبوابات التفريق والتجميع المتوازية لحالة الاستخدام الاولى

من خلال الجدول السابق (4.1) نلاحظ تطابق بين البوابتين (ps1,pm2) وعدم وجود تطابق بين البوابتين (ps2,pm1)، وطالما أنه يوجد رابط مباشر بين هاتين البوابتين، فيمكن اعتبار الرابط بين البوابتين (**ps2,pm1**) هو رابط متزامن.

سنقوم الان بحساب العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة لإجرائية تقديم القرض والنتيجة موضحة ضمن الجدول (4.2). نتيجة الخوارزمية هو توصيف مهيكل للإجرائية باستخدام نحو اللغة التنفيذية الموصف في القسم (3.7)، حيث يتم توصيف الإجرائية بالتوصيف التالي:

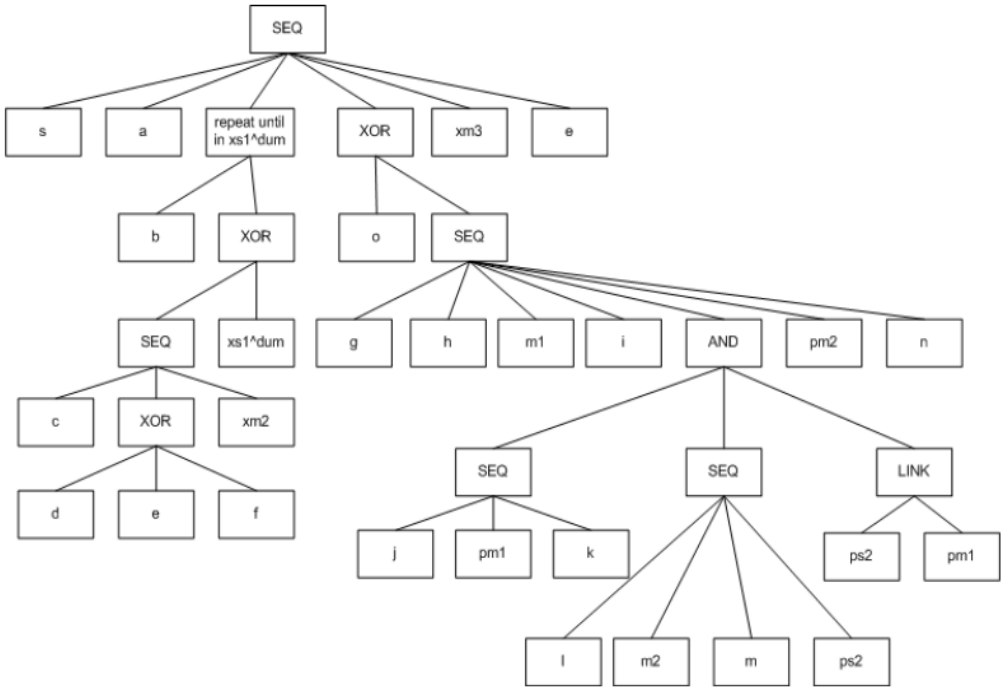
P=<s,a,repeat P_x until in (xs_1^{dum}), xor{o,<g,h, m_1 ,i, and{< j, pm_1 ,k >, <l,m_2,m,ps_2>,{link(ps_2, pm_1)}},pm_2,n >},xm_3,e>

p_x = < b, xor{xs_1^{dum}, <c, xor{d,e,f},xm_2 >} >

وهذه الإجرائية موصـــفة بالتراكيب المهيكلة في الشـــكل (4.2)، فالتراكيب المهيكلة تمثل أفضـــل حل نرغب فيه للتحويل باستثناء بوابات الدمج (merge-gateways)، فالتراكيب المهيكلة لا تحتاج مثل هذه البوابات وذلك لأنها لا تمثل نشاط، فيمكن حذف العقد (xm2, pm2, xm3).

العقدة	العقدة المهيمنة	رأس الحلقة	المجموعات اللاحقة
s	-	-	a
a	s	-	xm1
xm1	a	Loopheader	xs_1^{dum}
b	xm1	xm1	xs1
xs1	b	xm1	c
c	xs1	xm1	xs2
xs2	c	xm12	xm2
d	xs2	xm1	–
e	xs2	xm1	–
f	xs2	xm1	–
xm2	xs2	xm1	xm1
g	xs_1^{dum}	-	h
h	g	-	m1
m1	h	–	i
i	m1	-	ps1
ps1	i	–	pm2
j	ps1	–	pm1
pm1	j	–	k
k	pm1	–	–
l	ps1	–	m2
m2	l	–	m
m	m2	–	ps2
ps2	m	–	–
pm2	ps1	–	n
n	pm2	–	–
o	xs_1^{dum}	-	-
xm3	xs_1^{dum}	-	e
e	xm3	–	–

الجدول (4.2) العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة لحالة الاستخدام الأولى

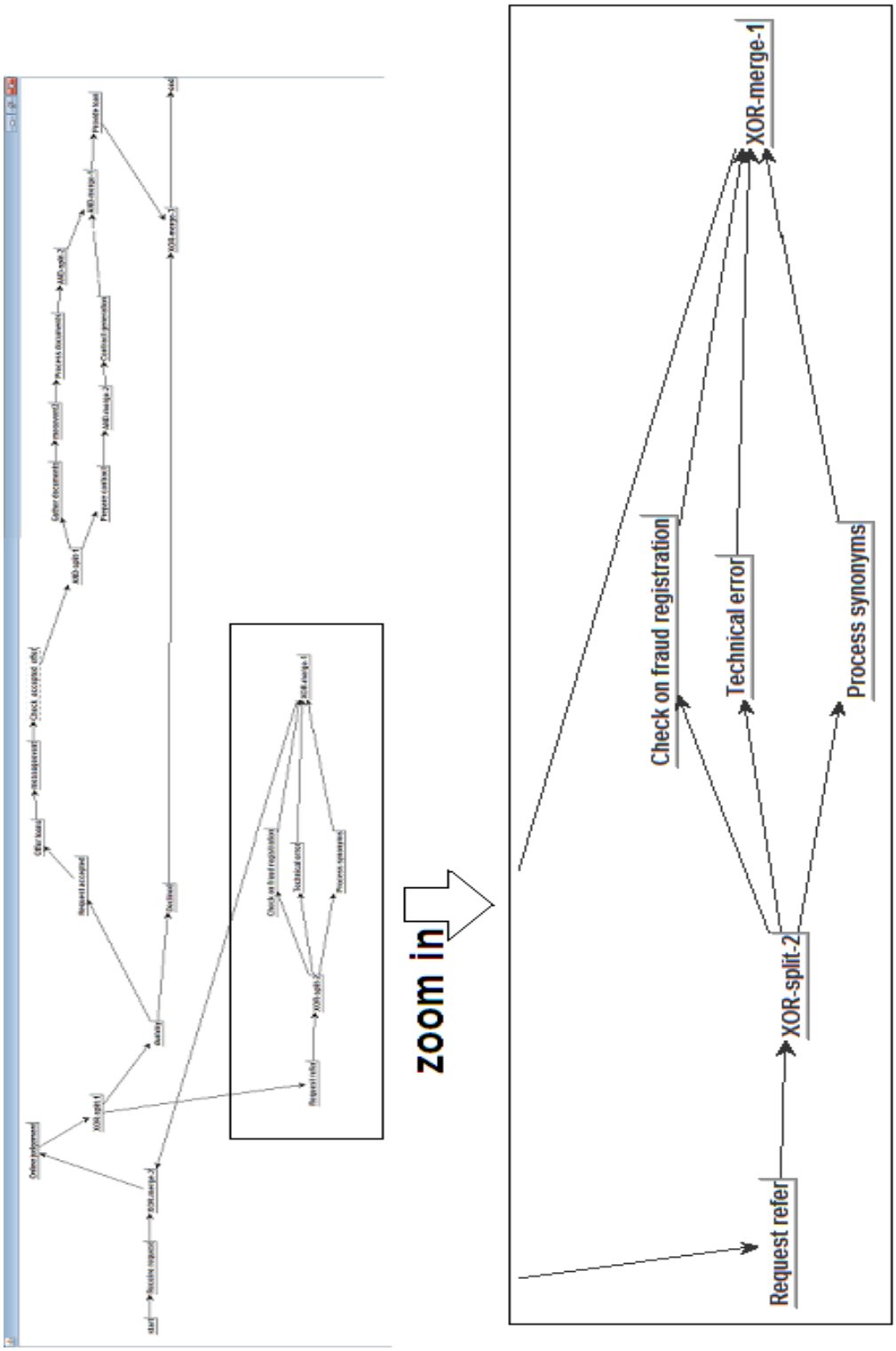


الشكل (4.2) التركيب المهيكل للحالة المدروسة الأولى

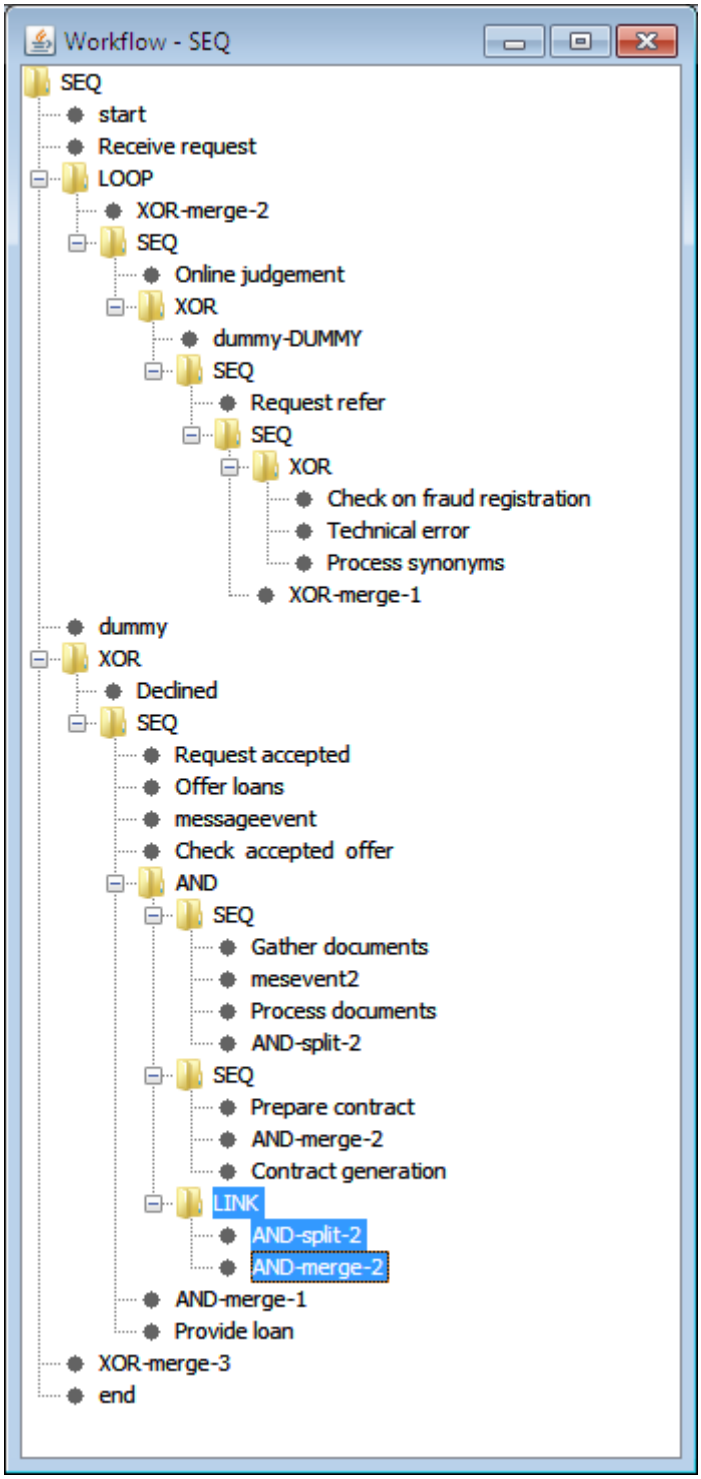
4.1.2. التحويل باستخدام التنجيز البرمجي

يقوم التنجيز البرمجي بتحويل مخطط إجرائية العمل إلى بيان تبعيات الموضح في الشكل (4.3)، ليتم بعدها تحويل هذا البيان إلى شجرة الإجرائية المهيكلة الموضحة في الشكل (4.4)، حيث نلاحظ الرابط المتزامن بين البوابتين (AND-split-2, AND-merge-2) والممثل بالعقدة (LINK). وليتم بعدها تحويل الشجرة السابقة إلى رماز لغة تنفيذية يمثل الشكل (4.5) جزءا منه، وحيث يظهر الرماز الممثل للرابط المتزامن.

```
<links>
        <link name="AND-split-2-to-AND-merge-2" />
</links>
```



الشكل (4.3) بيان التبعيات الممثل للحالة المدروسة الأولى



الشكل (4.4) شجرة الإجرائية المهيكلة للحالة المدروسة الأولى

```
  <sequence>
    <invoke name="Online judgement" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficient
    <if min_duration="P0D">
      <invoke name="dummy-DUMMY" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficientFun
      <sequence>
        <invoke name="Request refer" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficien
        <sequence>
          <if min_duration="P0D">
            <invoke name="Check on fraud registration" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operat
            <invoke name="Technical error" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsuf
            <invoke name="Process synonyms" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsu
          </if>
          <invoke name="XOR-merge-1" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficien
        </sequence>
      </sequence>
    </if>
  </sequence>
</sequence>
<invoke name="dummy" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficientFundException"
<if min_duration="P0D">
  <invoke name="Declined" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficientFundExcept
  <sequence>
    <invoke name="Request accepted" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficient
    <invoke name="Offer loans" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficientFundE
    <invoke name="messageevent" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficientFund
    <invoke name="Check  accepted  offer" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsuff
    <flow>
      <links>
        <link name="AND-split-2-to-AND-merge-2" />
      </links>
      <sequence>
```

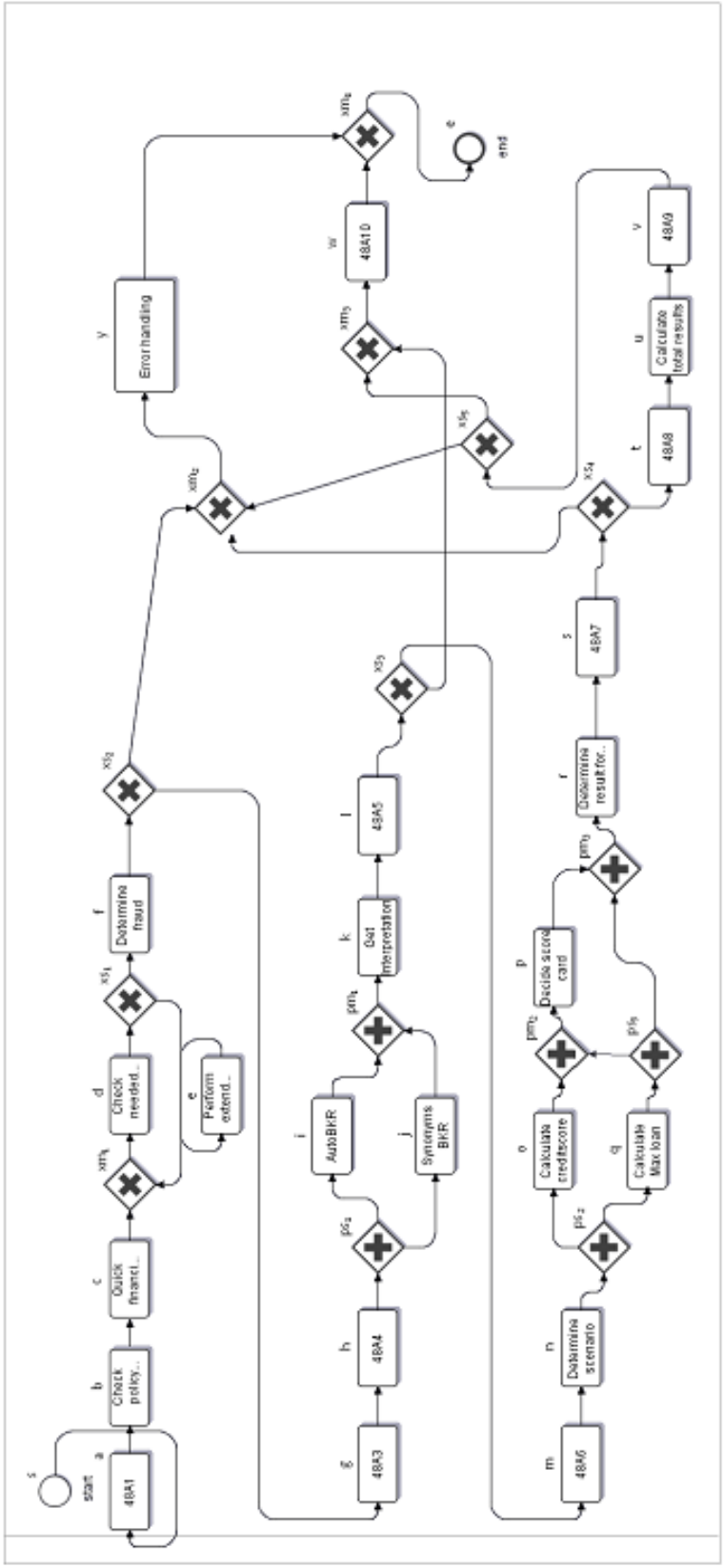
الشكل (4.5) جزء من رماز اللغة التنفيذية للحالة المدروسة الأولى

4.2. الحالة المدروسة الثانية: حساب المخاطر الكلية لبطاقة الائتمان

الحالة المدروسة الثانية التي سنناقشها تمثل مخطط إجرائية عمل لحساب المخاطر الكلية لبطاقة الائتمان، هذه الإجرائية تشكل طبقة وسيطة بين تطبيقات المستخدم النهائي والخدمات الوظيفية المنجزة من قبل الشركة، وهذه الإجرائية ممثلة بمخطط إجرائية العمل في الشكل (4.6).

يتم تقديم طلب حساب المخاطر عل بطاقة الائتمان من قبل مستخدم عادي أو من قبل شركة تأمين. بعد تقديم طلب فإنه يجري استدعاء مجموعة من الأنشطة بشكل متسلسل، بداية يتم طلب النشاط الداخلي (A.148) ليتم بعدها اختبار تحقق شروط وثيقة التأمين عن طريق استدعاء نشاط اخر. ثم يجري اختبار مالي سريع، وبنتيجة هذا الاختبار يتم تحديد إذا كان الطلب قد استوفى كافة المعلومات الضرورية، ويتم طلب الاختبار في حال نقص المعلومات، وتدخل في حلقة الاستدعاء حتى يتم استيفاء كافة المعلومات. وبعد استكمال كافة المعلومات، يجري تقييم الطلب وتحديد مخاطر الاحتيال. وفي حال كانت المخاطر عالية تقوم الإجرائية بإعادة خطأ. وإلا يتم استدعاء المزيد من الأنشطة الداخلية التي تقوم بالتحقق من شدة أمان البطاقة.

يتم استدعاء الخدمتين (AutoBKR, SynonymsBKR) بشكل متواز، ويتم تجميع نتيجة الاستدعاء بعد انتهاء كلتا الخدمتين. ويقوم النشاط (48A.5) بتقييم نتيجة الاستدعاء السابق ويقرر فيما إذا كنا بحاجة إلى مزيد من الاختبار أو يعطي النتيجة إلى النشاط الداخلي (48A.10). وفي حالة كنا بحاجة إلى المزيد من الاختبار فيتم الدخول بسيناريو جديد ويكون التزامن بين إجرائيتين أُخريين. يتم تحديد درجة نقاط بطاقة الائتمان وحساب الحد الأعظمي للقرض، فبعد حساب نقاط الائتمان، يتم تحديد بطاقة النقاط. ولكن هذا يحتاج أيضا إلى معرفة القيمة العظمى للقرض، وبالتالي يفترض وجود رابط متزامن ضمن النموذج. يتم تحديد نتيجة السيناريو، أو يمكن أن ينتهي السيناريو بدون نتيجة بسبب رسالة خطأ، فنتيجة السيناريو تكون أما قيمة موجبة أو خطأ. فالنتيجة الموجبة يتم معالجتها من قبل نشاط داخلي أخر (48A.10) ويتم إعادتها إلى مستخدم الإجرائية.



الشكل (4.6) الحالة المدروسة الثانية: حساب المخاطر الكلية لبطاقة الإئتمان

4.2.1. التحويل اليدوي

سنقوم الان بإجراء التحويل اليدوي للحالة المدروسة والمشروحة سابقا إلى تركيب مهيكل ممثلا بإجرائيات اللغة التنفيذية. ففي البداية سيتم تطبيق خوارزمية إيجاد الروابط المتزامنة، ومن ثم تطبيق خوارزمية التحويل والحصول على التركيب المهيكل، ليتم بعدها تقييم خرج التحويل ومقارنته مع نتيجة التحويل باستخدام التنجيز البرمجي المقدم في الفصل السابق.

قبل تطبيق خوارزمية إيجاد الروابط المتزامنة، سنقوم بتحديد الروابط المباشرة بين بوابات التفريق المتوازية وبوابات التجميع المتوازية، ففي حال عدم وجود روابط مباشرة فليس هناك حاجة لتطبيق الخوارزمية باعتبار وجود رابط مباشر بين بوابة التفريق المتوازية وبوابة التجميع المتوازية شرط أساسي لوجود رابط متزامن.

والجدول (4.3) يحوي المجموعتين before(g)-set,after(g)-set لبوابات التفريق والتجميع المتوازية.

البوابة	before(g)-set	after(g)-set
ps1	---	i, j, pm1
ps2	---	o, pm2, p, q, ps3, pm3
ps3	ps2, q	pm2, p, pm3
pm1	ps1, i, j	---
pm2	ps2, o, q, ps3	p, pm3
pm3	ps2, o, pm2, p, q, ps3	---

الجدول (4.3) المجموعتين before(g)-set, after(g)-set

لبوابات التفريق والتجميع المتوازية للحالة المدروسة الثانية

من خلال الجدول السابق نلاحظ وجود تطابق بين البوابتين (ps1, pm1) وكذلك يوجد تطابق بين البوابتين(ps2, pm3)، بينما لا يوجد تطابق بين البوابتين (ps3, pm2). وبسبب وجود رابط مباشر بين هاتين البوابتين فيمكن اعتبار الرابط بين البوابتين (ps3, pm2) هو رابط متزامن.

سنقوم بتحديد العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة للحالة المدروسة والموضحة بالجدول (4.4).

العقدة	العقدة المهيمنة	رأس الحلقة	المجموعات اللاحقة
s			a
a	s		b
b	a		c
c	b		xm1
xm1	c	loopnode	f
d	xm1	xm1	xs1
xs1	d	xm1	–
e	xs1	xm1	–
f	xs1		xs2
xs2	f		xm2; xm4
g	xs2		h
h	g		ps1
ps1	h		pm1
i	ps1		–
j	ps1		–
pm1	ps1		k
k	pm1		l
l	k		xs3
xs3	l		xm3
m	xs3		n
n	m		ps2
ps2	n		pm3
o	ps2		pm2
pm2	o		p
p	pm2		–
q	ps2		ps3
ps3	q		–
pm3	ps2		r
r	pm3		s
s	r		xs4
xs4	s		–
t	xs4		u
u	t		v
v	u		xs5
xs5	v		–
xm2	xs2		y
xm3	xs3		w
w	xm3		–
y	xm2		–
xm4	xs2		e
e	xm4		–

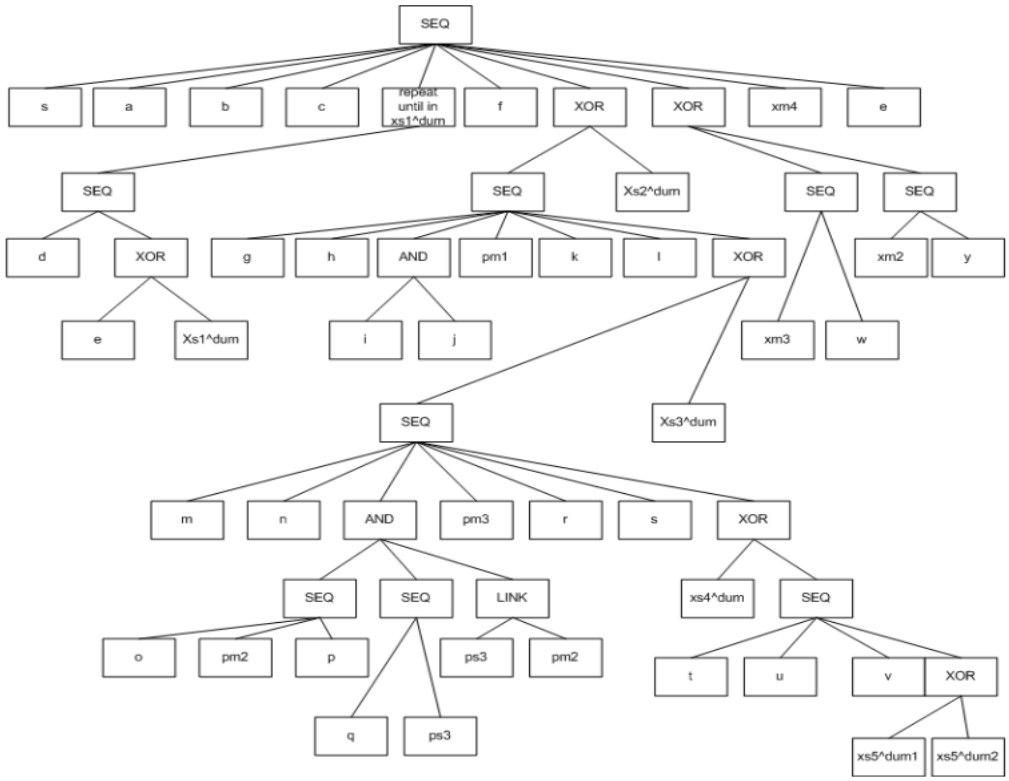
الجدول (4.4) العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة للحالة المدروسة الثانية

وينتج عن تطبيق الخوارزمية السابقة التوصيف المهيكل وفق نحو اللغة التنفيذية المعرفة ضمن القسم (3.4)

$$P =< s, a, b, c, \text{repeat } P_x \text{ until in } (xs_1^{dum}), f, \text{xor}\{P_y, xs_2^{dum}\}, xm_2, y, xm_4, e >$$

$$P_x =< d, \text{xor}\{e, xs_1^{dum}\} >$$
$$P_y =< g, h, \text{and}\{i, j\}, pm_1, k, l, \text{xor}\{P_z\}, xm_3, w >$$
$$P_z =< xs_3^{dum}, < m, n, \text{and}\{< o, pm_2, p >, < q, ps_3 >, \{\text{link}(ps_3, pm_2)\}\}, pm_3, r, s,$$
$$\text{xor}\{xs_4^{dum}, < t, u, v, \text{xor}\{xs_5^{dum1}, xs_5^{dum2}\} >\} >>$$

وهذه الإجرائية موصفة بالتركيب المهيكل في الشكل (4.7)، فالتراكيب المهيكلة تمثل أفضل حل نرغب فيه للتحويل باستثناء بوابات الدمج (merge-gateways)، فالتراكيب المهيكلة لا تحتاج مثل هذه البوابات وذلك لأنها لا تمثل نشاط، فيمكن حذف العقد (pm1, pm3, xm2, xm3, xm4).

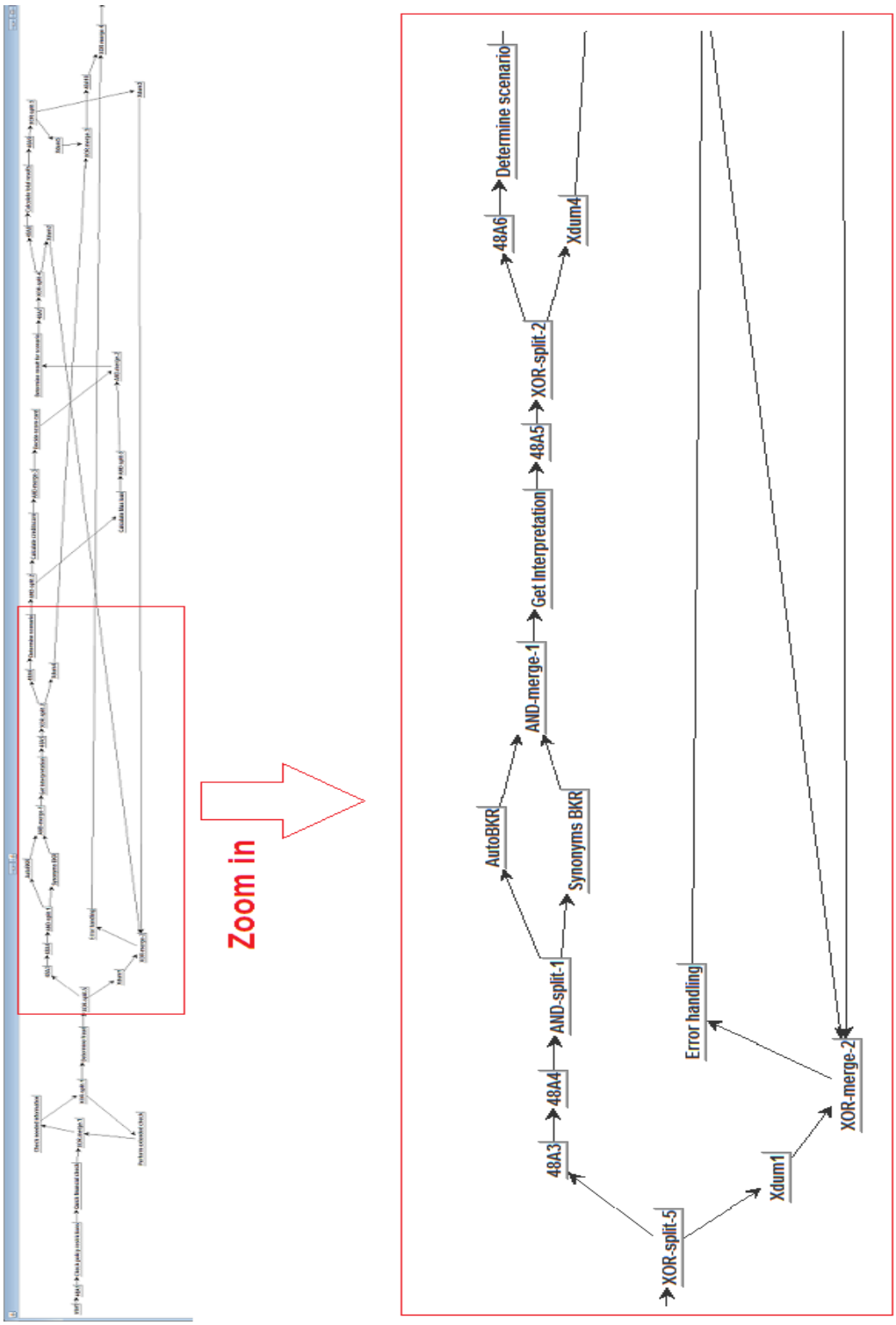


الشكل (4.7) التركيب المهيكل للحالة المدروسة الثانية

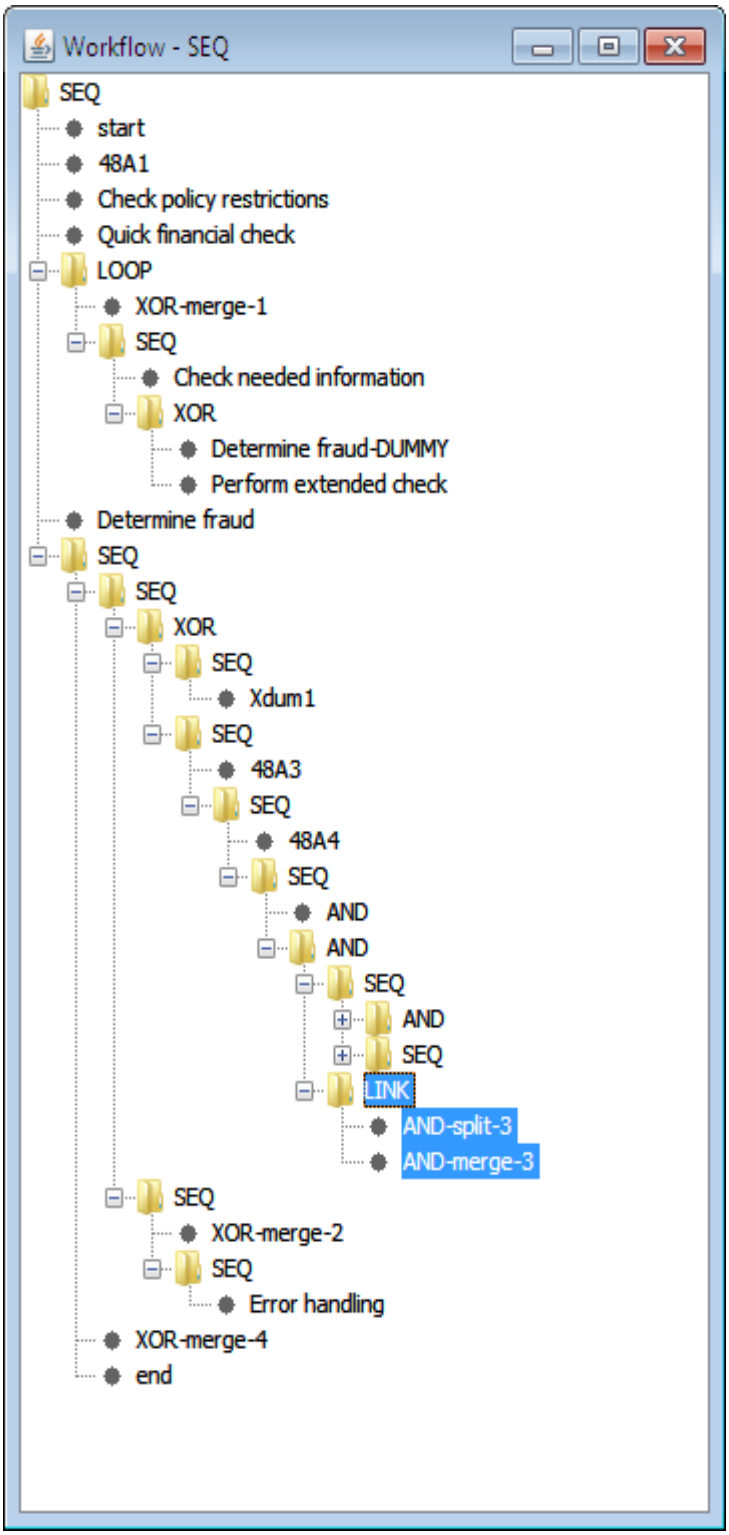
4.2.2. التحويل باستخدام التنجيز البرمجي

وبنفس طريقة الحالة المدروسة الأولى يقوم التنجيز البرمجي بتحويل مخطط إجرائية العمل إلى بيان تبعيات الموضح في الشكل (4.8)، ليتم بعدها تحويل هذا البيان إلى شجرة الإجرائية المهيكلة الموضحة في الشكل (4.9)، حيث نلاحظ الرابط المتزامن بين البوابتين (AND-split-3, AND-merge-3) والممثل بالعقدة (LINK). وليتم بعدها تحويل الشجرة السابقة إلى رماز لغة تنفيذية يمثل الشكل (4.10) جزءا منه، وحيث يظهر الرماز الممثل للرابط المتزامن.

```
<links>
        <link name="AND-split-3-to-AND-merge-3" />
</links>
```



الشكل (4.8) بيان التبعيات الممثل للحالة المدروسة الثانية



الشكل (4.9) الإجرائية المهيكلة للحالة المدروسة الثانية

```
<invoke name="Determine fraud" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficientFundEx
<sequence>
  <sequence>
    <if min_duration="P0D">
      <sequence>
        <invoke name="Xdum1" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficientFundExce
      </sequence>
      <sequence>
        <invoke name="48A3" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficientFundExcep
        <sequence>
          <invoke name="48A4" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsufficientFundExc
          <sequence>
            <flow />
            <flow>
              <links>
                <link name="AND-split-3-to-AND-merge-3" />
              </links>
              <sequence>
                <flow>
                  <sequence>
                    <invoke name="Synonyms BKR" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="on
                  </sequence>
                  <sequence>
                    <invoke name="AutoBKR" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsuf
                  </sequence>
                </flow>
                <sequence>
                  <invoke name="AND-merge-1" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operation="onInsu
                  <sequence>
                    <invoke name="Get Interpretation" partnerLink="client" portType="tns:PaymentProcessorServiceCallback" operatio
                    <sequence>
```

الشكل (4.10) جزء من رماز اللغة التنفيذية للحالة المدروسة الثانية

الفصل الخامس

تقييم نموذج التحويل

5. تقييم نموذج الحل

سنقوم ضمن هذا القسم بتقييم نموذج التحويل الذي قمنا ببنائه في الفصل الثالث، حيث سنذكر نقاط القوة ونقاط الضعف لنموذج التحويل وسنقوم بمقارنته مع المنهجيات والتنجيزات البرمجية السابقة. وباعتبار أنَّ الخوارزمية المقترحة تتطلب مخططات إجرائية عمل مبنية بشكل جيد، فسنناقش شروط البناء الجيد وإمكانية التراخي ببعض هذه الشروط وانعكاسه على عملية التحويل.

5.1. نقاط القوة والضعف

لقد ناقشنا في القسم الثاني عدة استراتيجيات وحددنا نقاط القوة والضعف لكل منهجية، سنقوم في (5.2) بمقارنة نموذج التحويل الذي قمنا ببنائه مع هذه الاستراتيجيات، لكن بداية سنقوم بذكر نقاط القوة والضعف الرئيسية للخوارزمية المقترحة.

تدعم الخوارزمية المطروحة النماذج التي تملك بعض المكونات غير المهيكلة باستثناء الحلقات الكيفية، أي أنها لا تعالج فقط النماذج المهيكلة وإنما تعالج النماذج غير المهيكلة بشرط تحقيقها لشروط البناء الجيد. فالنموذج يقبل جميع البوابات المتوازية. كما أن رماز اللغة التنفيذية المتولد مفهوم وقابل للقراءة إذ أنه مبني ومهيّكل بشكل جيد، وتم توسيع رماز اللغة التنفيذية الناتج ليشمل تبعيات التزامن وهي مفهومة أيضا باعتبارها تستخدم فقط لأغراض التزامن.

يتألف نموذج الحل من عدة خطوات إذا يبدأ بمرحلة التحضير حيث يتم تحديد وحساب بعض المفاهيم كالعقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة، ومن ثم يجري تنفيذ الخوارزمية الرئيسية والتحويل إلى رماز اللغة التنفيذية. والرماز الناتج يكون قابل للقراءة ومفهوم باعتباره ناتج عن تحويل بنى مهيكلة هي نتيجة تطبيق خوارزمية إيشويس المعدَّلة من قبلنا والتي تقوم بتحويل التراكيب غير المهيكلة إلى تراكيب مهيكلة.

5.2. المقارنة مع باقي المنهجيات والبرمجيات

سنقوم في البداية بمقارنة النموذج المقترح مع المنهجيات المطروحة في القسم الثاني، فرماز اللغة التنفيذية الناتج عن النموذج المقترح **أكثر قابلية للقراءة** من الرماز الناتج عن تطبيق الخوارزميات المعتمدة على الاحداث والموضحة ضمن الجدول (2.4)، فالمكونات المهيكلة يجري تحويلها إلى بنى اللغة التنفيذية المقابلة لها من رماز اللغة التنفيذية وبالتالي فنموذج اللغة التنفيذية الناتج يحافظ على وضوحه وسهولة قراءته. وكذلك فإن مفهوم البنى التكرارية كالحلقة غير مدعومة في الاستراتيجيات النقية (غير الهجينة) المبنية على الرابط. بينما النموذج المقترح يدعم الحلقات ذات نقطة الدخل واحدة ونقطة الخرج واحدة، وكذلك يدعم جميع أنواع الروابط المتزامنة بما فيها الروابط المتزامنة بين الفروع الناتجة عن تزامنات سابقة. بمقارنة النموذج المقترح مع الاستراتيجيات المعتمدة على البنية فإن هذا النموذج يقبل مخططات دخل بشروط أدق ويدعم المخططات غير المهيكلة وبذلك فهو يتفوق على الخوارزميات التي يكون خرجها مفهوماً.

أي أن النموذج المقترح **يتفوق بقابلية قراءة رماز الخرج** على الخوارزميات التي تعتبر قريبة من الكمال من ناحية تحقيق العدد الأكبر من نماذج الدخل (أ، ب، ج، خ، ذ) في الجدول (2.4)، **ويتفوق بالكمال** على الخوارزميات ذات الخرج القابل للقراءة (ت، ث، ح، د، ر) في الجدول (2.4) من ناحية معالجته للنماذج التي تحوي بعض البنى غير المنفذة بتلك الخوارزميات كالحلقات المهيكلة والروابط المتزامنة.

وبالنسبة للتنجيز البرمجي لنموذج الحل فسنقارنه مع الأداتين البرمجيتين المتوفرتين (BABEL-tool, BPMN2BPEL Eclipse plugin) المذكورتين في القسم الثاني من الأطروحة. **فيتميز** التنجيز البرمجي للنموذج المقترح عن التنجيزات الأخرى بخوارزمية إيجاد التبعيات المتزامنة فيتم تحويل المكونات المهيكلة التي تحتوي على تبعيات متزامنة بنفس الطريقة مما يحافظ على قابلية قراءة رماز اللغة التنفيذية الناتج عن عملية التحويل، بينما تقوم التنجيزات الأخرى بتحويل هذه المكونات المهيكلة كمكونات غير مهيكلة ومما يؤثر سلبا على قابلية قراءة رماز اللغة التنفيذية الناتج.

جميع التنجيزات البرمجية تدعم الحلقات لكن التنجيز البرمجي للحل المقترح يدعم فقط الحلقات المهيكلة ذات نقطة الدخل الواحدة ونقطة الخرج الواحدة، بينما التنجيزين الآخرين يقومان بتحويل الحلقات غير المهيكلة باستخدام الاستراتيجيات المبنية على حدث–فعل غير المنجز في تحويلنا.

سنقوم الان بتقييم التنجيزات المتوافرة من خلال تحويل الحالتين المدروستين الموصفتين سابقا واللتين قمنا بتحويلهما باستخدام النموذج المقترح يدويا وباستخدام تنجيزه البرمجي. إن التنجيز البرمجي للحل المقترح يستخدم كدخل نماذج تمثيل رمزي BPMN مبنية باستخدام Eclipse. بينما الأداة BABEL تستخدم شكل دخل مختلف لتوصيف الإجرائيات. لذلك فإننا نحتاج لوجود الحالتين المدروستين بكلا الشكلين لنتمكن من تجربتهما.

سنقوم الأن بمناقشة الحالتين المدروستين بشكل منفصل.

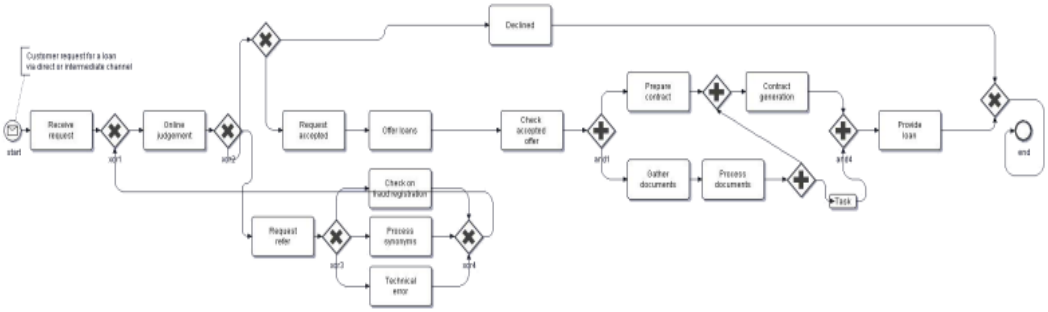
5.2.1. الحالة المدروسة الأولى

كما رأينا في القسم السابق تم تحويل الحالة المدروسة الأولى باستخدام الخوارزمية المقترحة وتنجيزها البرمجي وحصلنا على نموذج الخرج المتوقع، ولم نكن بحاجة إلى تعديل نموذج الدخل لنحصل على تحويل صحيح. بينما نواجه بعض المشاكل في تحويل الحالتين المدروستين باستخدام الأداتين البرمجيتين الآخرين.

- فالإضافة (BPMN2BPEL) تواجه مشاكل بخصوص الأحداث الوسيطة في النموذج الأصلي لحالة الاستخدام وتعمل فقط بعد تجزئة بوابة القرار الأولى يدويا، هذه الإضافة تدعم التبعيات المتزامنة لكن يتم تحويل التدفق المتوازي كمكون غير مهيكل مما يؤثر سلبا على قابلية القراءة رماز اللغة التنفيذية الناتج. كذلك فإن التحويل يستخدم بنى فارغة بدلا من بنية الاستدعاء. والشكل (6.1) يمثل جزءاً من رماز اللغة التنفيذية المتولد عن تحويل نموذج التمثيل الرمزي ضمن حالة الاستخدام الأولى بعد تعديله ليصبح كما في الشكل (5.2)

```
                        <bpws:empty name="Technical&#xd;&#xa;error"/>
                    </bpws:elseif>
                </bpws:if>
            </bpws:sequence>
            <bpws:empty name="Online&#xd;&#xa;judgement"/>
        </bpws:sequence>
    </bpws:while>
</bpws:sequence>
<bpws:if name="null-null">
    <bpws:empty name="Declined"/>
    <bpws:elseif>
        <bpws:sequence name="Request&#xd;&#xa;accepted-Provide&#xd;&#xa;loan">
            <bpws:empty name="Request&#xd;&#xa;accepted"/>
            <bpws:empty name="Offer loans"/>
            <bpws:empty name="Check &#xd;&#xa;accepted &#xd;&#xa;offer"/>
            <bpws:flow name="and1-and4">
                <bpws:links>
                    <bpws:link name="null__null"/>
                    <bpws:link name="Process&#xd;&#xa;documents__null"/>
                    <bpws:link name="Process&#xd;&#xa;documents__and4"/>
                    <bpws:link name="null__Contract&#xd;&#xa;generation"/>
```

الشكل (5.1) الحالة المدروسة الأولى بعد تحويلها باستخدام الإضافة (BPMN2BPEL)

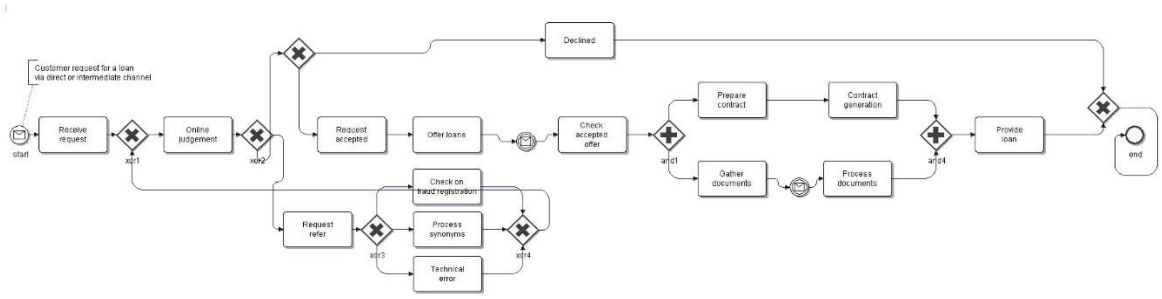


الشكل (5.2) نموذج الحالة المدروسة الأولى بعد تعديله ليناسب الإضافة (BPMN2BPEL)

- كذلك الأمر بالنسبة للأداة BABEL فيمكنها تحويل نموذج الدخل في حالة الاستخدام الأولى ولكن بعد إجراء بعض التعديلات البسيطة عليه، فالتبعيات المتزامنة غير مدعومة ضمن الأداة ونتائج مختلفة تتولد طبقا للبوابات الوهمية التي تتم إضافتها إلى نموذج الدخل. فعند إضافة البوابات الوهمية إلى نموذج الدخل نحصل على رماز اللغة التنفيذية كما في الشكل (5.3)، وفي حالة عدم إضافة بوابات وهمية فإنه يتم تحويل كل الإجرائيات بعد البوابة باستخدام المنهج المعتمد على الفعل وبالتالي فإن الخرج يكون أقل قابلية للقراءة، والشكل المستخدم كدخل للأداة BABEL موضح ضمن الشكل (5.4)

```
<sequence name="sequenceComponent_5">
  <receive name="ProcessInstantiation" partnerLink="client" portType="localPT" operation="localPT" variable="client_data" createIns
  <invoke name="register" partnerLink="local" portType="localPT" operation="register" inputVariable="register_data_in" outputVariab
  <flow name="flowComponent_4">
    <sequence name="sequenceComponent_2">
      <invoke name="sendQuestionnaire" partnerLink="local" portType="localPT" operation="sendQuestionnaire" inputVariable="sendQues
      <pick name="pickComponent_1">
        <onMessage name="returnedQuestionaire" partnerLink="local" portType="localPT" operation="returnedQuestionaire">
          <invoke name="processQuestionnaire" partnerLink="local" portType="localPT" operation="processQuestionnaire" inputVariable
        </onMessage>
        <onAlarm name="timeOut" partnerLink="local" portType="localPT" operation="timeOut" />
      </pick>
    </sequence>
    <scope name="NWC_3">
      <eventHandlers>
        <onEvent name="Start(NWC_3)&#xA;" partnerLink="local" portType="localPT" operation="Start(NWC_3)&#xA;">
          <invoke name="invoke end(c2)" partnerLink="local" portType="localPT" operation="invoke end(c2)" />
        </onEvent>
        <onEvent name="switch(c4,c2)&#xA;" partnerLink="local" portType="localPT" operation="switch(c4,c2)&#xA;">
          <invoke name="invoke end(c2)" partnerLink="local" portType="localPT" operation="invoke end(c2)" />
        </onEvent>
```

الشكل (5.3) الحالة المدروسة الأولى بعد تحويلها باستخدام الإضافة (BABEL)



الشكل (5.4) مخطط إجرائية عمل الحالة المدروسة الأولى بعد تعديله ليناسب الإضافة (BABEL)

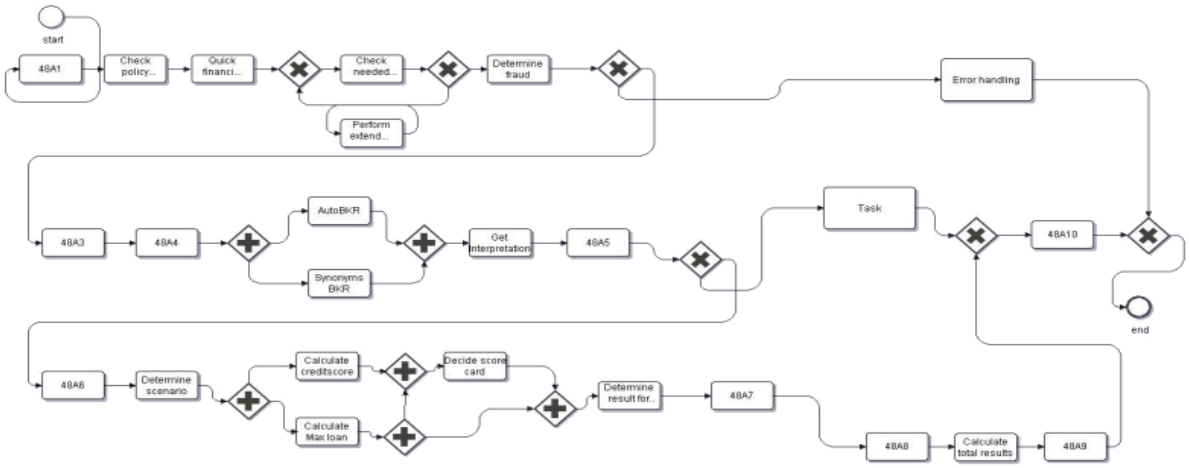
5.2.2. الحالة المدروسة الثانية

كما رأينا سابقا فإنه جرى تحويل حالة الاستخدام الثانية باستخدام النموذج المقترح ولاحظنا ظهور المشاكل بعملية التحويل بسبب وجود بوابات الدمج المتعدد، وكذلك الأمر فإن الاداتين الأخريين تعانيان من المشاكل أيضاً عند تحويل نموذج حالة الاستخدام الثانية كما سنرى.

- فالإضـافة (BPMN2BPEL) تعاني من المشـاكل بسـبب وجود بوابات الدمج والقرار المتعدد. لقد قمنا بحذف بوابة دمج واحدة وبوابة توزيع واحدة وقمنا بإضافة عقدة وهمية بين البوابات المتبقية والتي تم وصلها مباشـرة. إن هذه الإضـافة تدعم التبعيات المتزامنة ولكن يتم معاملتها كمكونات غير مهيكلة مما يضـعف قابلية القراءة لرماز اللغة التنفيذية الناتج عن التحويل. والشـكل (5.5) يوضـح جزء من رماز اللغة التنفيذية الناتج عن نموذج حالة الاسـتخدام الثانية بعد التعديل ليناسـب الإضـافة (BPMN2BPEL) والموضح ضمن الشكل (5.6).

```
<?xml version="1.0" encoding="UTF-8"?>
<bpws:process exitonStandardFault="yes" name="Case study 2"
    suppressJoinFailure="yes" targetNamespace="http://tempuri.org/plks"
    xmlns:bpws="http://docs.oasis-open.org/wsbpel/2.0/process/executable" xmlns:tns="http://tem
    <bpws:sequence name="start-end">
        <bpws:empty name="start"/>
        <bpws:empty name="48A1"/>
        <bpws:empty name="Check&#xd;&#xa;policy&#xd;&#xa;restrictions"/>
        <bpws:empty name="Quick financial check"/>
        <bpws:sequence name="null-null_firstIter">
            <bpws:empty name="Check needed information"/>
            <bpws:while name="null-null">
                <bpws:sequence>
                    <bpws:empty name="Perform extended check"/>
                    <bpws:empty name="Check needed information"/>
                </bpws:sequence>
            </bpws:while>
        </bpws:sequence>
        <bpws:empty name="Determine fraud"/>
        <bpws:if name="null-null">
```

الشكل (5.5) حالة الاستخدام الثانية بعد تحويلها باستخدام الإضافة (BPMN2BPEL)



الشكل (5.6) نموذج الحالة المدروسة الثانية بعد تعديله ليناسب الإضافة (BPMN2BPEL)

- كما أن الأداة (BABEL) تعاني أيضا من مشاكل مع النموذج الأصلي للحالة المدروسة الثانية، وحتى بعد حذف التبعيات المتزامنة كما فعلنا في الحالة المدروسة الأولى فإننا لم نستطيع إجراء التحويل باستخدام الأداة (BABEL) حيث بقيت المشاكل بسبب وجود بنى التدفق المتوازية.

بسبب وجود التبعيات المتزامنة، وحتى بعد حذف التبعيات المتزامنة كما فعلنا في حالة الاستخدام الأولى فإننا لم نستطيع إجراء التحويل باستخدام الأداة (BABEL) حيث بقيت المشاكل بسبب وجود بنى التدفق المتوازية.

5.2.3. الخاتمة

نقوم بتجميع النتائج التي حصلنا عليها في المناقشة السابقة ضمن الجدول (5.1)، حيث يلخص ميزات وامكانيات الأدوات البرمجية المتوفرة للتحويل بين التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل ومقارنتها مع التنجيز البرمجي لنموذج التحويل الخاص بنا.

	BABEL	BPMN2BPEL	النموذج المقترح
قابلية القراءة	+	+/−	+
المكونات المهيكلة	+	+	+
الحلقات المهيكلة	+	+	+
الحلقات غير المهيكلة	+	−	−
الروابط المتزامنة	−	+/−	+

الجدول (5.1) المقارنة بين النموذج المقترح والأداتين (BABEL, BPMN2BPEL)

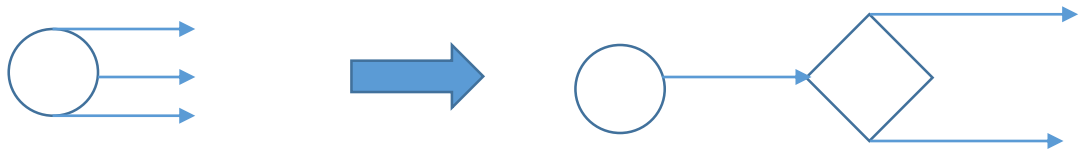
يمكن تحسين الإضافة (BABEL) عن طريق إضافة خوارزمية إيجاد التبعيات المتزامنة وذلك بهدف الحصول على رماز لغة تنفيذية أكثر قابلية للقراءة. ويمكن دعم هذه الإضافة بخوارزمية إيجاد التبعات المتزامنة الموجودة ضمن النموذج المقترح.

5.3. مناقشة شروط البناء الجيد لمخطط إجرائية العمل

كما رأينا سابقا فإنه يتوجب على نماذج BPMN المستخدمة كدخل لعملية التحويل بين التمثيل الرمزي واللغة التنفيذية أن تحقق شروط البناء الجيد لمخطط إجرائية العمل وهي ثمانية شروط موضحة ضمن التعريف 1 في القسم الثالث من الأطروحة. وقد تم بناء النموذج المقترح وفقا لهذه الشروط وبالتالي فإن النماذج التي لا تحقق هذه الشروط من الممكن ألا يتم تحويلها بشكل صحيح. لقد كان هدفنا في البداية بناء نموذج يقوم بتحويل كافة نماذج التمثيل الرمزي بما فيها النماذج غير المهيكلة، لكن بوجود شروط البناء الجيد للنموذج فليس هناك مجال لمعالجة النماذج غير المهيكلة، وبالتالي سنناقش الآن إمكانية الاستغناء عن بعض هذه الشروط لنموذج الدخل.

ix. حدث البداية يملك نقطة خرج واحدة ولا يملك نقطة دخل:

يتكون هذه الشرط من شرطين أثنين، الأول هو ألا يملك حدث البداية أي نقطة دخول وهذا الشرط لا يمكن الاستغناء عنه، فلو كان لحدث البداية نقطة دخل واحدة أو أكثر فهذه الوصلة هي قادمة من حدث أخر والذي يمكنه أن يبدأ الإجرائية وأصبح هو حدث البداية وكان حدثنا هو حدث وسيط وليس حدث بداية. أما القسم الثاني من الشرط فيمكن التراخي به، فيمكن إضافة بوابة وهمية تسمح بوجود أكثر من نقطة خرج لحدث البداية.

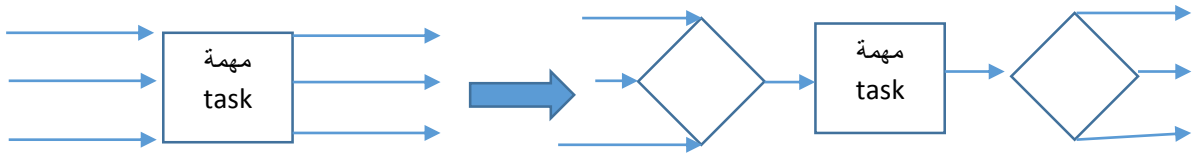


x. حدث النهاية يملك نقطة دخل واحدة ولا يملك نقطة خرج:

بنفس الكلام فإنه لا يمكن التخلي عن شرط عدم وجود نقط خرج لحدث النهاية، فإن وجود نقطة خرج لحدث النهاية يستلزم وجود حدث أخر لينهي الإجرائية. أما الجزء الثاني من الشرط (نقطة دخل واحدة) فيمكن التراخي بخصوصه عن طريق إضافة بوابة تجميع وهمية قبل حدث النهاية

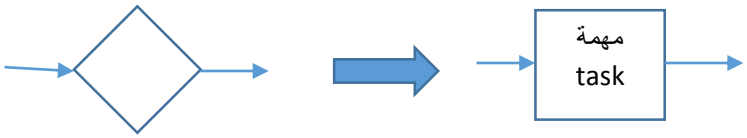
xi. المهام والاحداث الوسيطة تملك نقطة دخل واحدة ونقطة خرج واحدة:

لقد تم وضع خوارزمية تجميع الخدمات ضمن إجرائية مهيكلة [16] ليكون دخلها بيان تبعيات، حيث يتألف بيان التبعيات من شبكة من العقد والوصلات فالعقد تمثل المهام والوصلات تمثل العلاقات بين هذه المهام. يمكن للعقد أن تملك عدة وصلات داخلة وعدة وصلات خارجة لكن يتم معالجتها عن طريق إضافة عقدة تفريق وهمية أو عقدة حلقة وهمية، هذه العقد تمثل خدمات وهمية. وبالتالي يمكن السماح لوجود مهام أو أحداث وسيطة تملك أكثر من نقطة دخل أو أكثر من نقطة خرج، لكن بشرط معالجتها وإضافة عقدة وهمية قبل تطبيق خوارزمية التجميع.



xii. بوابات القرار والتفريق تملك نقطة دخل واحدة وعدة نقاط للخرج:

يجب على بوابات القرار والتفريق أن تملك أكثر من نقطة خرج، وإلا لن تكون عقدة تفريق أو قرار وإنما ستصبح عقدة عادية. أي أنه في حالة وجود بوابة تفريق أو قرار تملك نقطة خرج واحدة ونقطة دخل واحدة فإن الخوارزمية ستقوم بمعالجتها على أساس انها عقدة عادية وليست بوابة. وبالتالي يجب تحويل هذه البوابة إلى مهمة لتتم معالجتها بشكل صحيح ضمن الخوارزمية.



xiii. بوابات التجميع والدمج تملك عدة نقاط دخل ونقطة خرج واحدة:

بنفس طريقة الشرط السابق، فيمكن تحويل بوابة التجميع أو الدمج التي تملك نقطة دخل واحدة ونقطة خرج واحدة إلى مهمة أيضاً.

xiv. بوابة القرار XOR المبنية على الحدث يجب أن تتبع بأحداث وسيطة أو تستقبل مهام:

يتوجب على بوابة القرار XOR المبنية على الحدث أن تتبع بحدث وسيط أو أن تتلقى مهمة وإلا كانت بوابة مبنية على البيانات ويجب تصحيحها وتمثيلها كبوابة قرار مبنية على البيانات وليست بوابة مبنية على حدث.

xv. يجب أن يكون هناك مسار تدفق افتراضي لبوابات القرار المبني على البيانات:

ليس هناك ضرورة لتدفق افتراضي لبوابات القرار المبني على البيانات ضمن رماز اللغة التنفيذية، لكن يتوجب أن يكون هناك تدفق واحد على الأقل للسير به عند معالجة بوابات القرار المبني على البيانات، وبالتالي يتوجب على المصمم التأكد من وجود هكذا تدفق وعندها يمكن الاستغناء عن المسار الافتراضي لبوابة القرار. وبالتالي لا يمكن حذف هذا الشرط من الخوارزمية وإنما يستطيع المستخدم اهماله في حال تأكده من وجود مسار واحد على الأقل يمكن المرور به بعد معالجة البوابة.

xvi. كل غرض يجب أن يكون على مسار من حدث البداية وحتى حدث النهاية:

يجب أن يكون نموذج الدخل صحيح وتتم معالجة كافة مكونات النموذج. فالأغراض ضمن النموذج مرتبطة ببعضها ويتم معالجة هذه الأغراض والوصول إليها عن طريق حساب العقد المهيمنة ورؤوس الحلقات والمجموعات اللاحقة لكل غرض. فيتوجب علينا الوصول لكل غرض من النموذج انطلاقا من حدث البداية من خلال حساب المفاهيم السابقة لهذا الغرض، وأيضا كل غرض يجب أن يكون على مسار لحدث نهاية وذلك لعدم الوقوع بمشكلة القفل المميت (deadlock). وبالتالي لا يمكن التهاون بهذا الشرط.

- من خلال ما سبق نلاحظ أنه يمكننا التخلي عن معظم شروط البناء الجيد لمخطط إجرائية العمل، وذلك يكلفنا فقط بعض الخطوات الإضافية لمعالجة مخطط إجرائية العمل قبل البدء بتطبيق الخوارزمية كإضافة بعض العقد الوهمية وتحويل بعض البوابات إلى مهام. بينما توجد بعض الشروط لا يمكن التخلي عنها وشروط أخرى يمكن التخلي عن أجزاء منها.

الفصل السادس

الخاتمة والآفاق المستقبلية

6. الخاتمة والآفاق المستقبلية

سنقوم ضمن هذا القسم بتلخيص النتائج التي حصلنا عليها والمشاكل التي واجهتنا ومقترحاتنا للأعمال المستقبلية في مجال التحويل بين التمثيل الرمزي واللغة التنفيذية لإجرائيات العمل.

6.1. الخاتمة

ناقشنا ضمن هذه الأطروحة العديد من استراتيجيات تحويل التمثيل الرمزي لإجرائيات العمل إلى اللغة التنفيذية لإجرائيات العمل ووقفنا على نقاط الضعف والقوة لكل منهجية وقمنا بتطوير خوارزمية إيشويس [16] والتي تقوم بتحويل بيان التبعيات إلى إجرائيات مهيكلة، حيث قمنا بتحويل نموذج التمثيل الرمزي إلى بيان (عقد ووصلات) لنقوم بعدها بالبحث عن التبعيات المتزامنة ضمن البيان ومن ثم حساب بعض المفاهيم (العقد المهيمنة، رؤوس الحلقات والمجموعات اللاحقة) هذه المفاهيم الضرورية لتطبيق خوارزمية إيشويس، ليتم في نهاية الأمر تحويل هذه التراكيب المهيكلة إلى رماز لغة تنفيذية قابل للقراءة.

وبمقارنة النموذج المقترح مع المنهجيات السابقة في التحويل وجدنا أن النموذج المقترح يتفوق بقابلية قراءة رماز الخرج على الخوارزميات (أ، ب، ج، خ، ذ) – الجدول (2.4) والتي تعتبر قريبة من الكمال من ناحية قدرتها على تحويل العدد الأكبر من نماذج الدخل. ويتفوق أيضاً بالكمال على الخوارزميات (ت، ث، ح، د، ر) – الجدول (2.4) ذات الخرج القابل للقراءة وذلك بسبب قدرته على معالجة النماذج التي تحوي الحلقات المهيكلة والروابط المتزامنة غير المنفذة بتلك الخوارزميات.

أما مقارنة التنجيز البرمجي للنموذج المقترح مع التنجيزات البرمجية الأخرى، فإن هذا التنجيز غير قادر على تحويل الحلقات غير المهيكلة، ويتوجب على الحلقات أن تملك نقطة دخل واحدة ونقطة خرج واحدة. فالاستراتيجيات التي تسمح بمعالجة الحلقات غير المهيكلة تستخدم استراتيجيات مبنية على قاعدة حدث–فعل للتحويل والتي تؤدي إلى صعوبة قراءة رماز اللغة التنفيذية الناتج. بينما النموذج المقترح لا يستخدم هذه القواعد وإنما قمنا بتطوير خوارزمية لاستكشاف الروابط المتزامنة وحذفها مؤقتا من النموذج وبهذه الطريقة نستطيع معالجة المكونات غير المهيكلة كمكونات مهيكلة. والجدول (5.1) يمثل ملخص للمقارنة بين

التنجيز البرمجي للحل المقترح والأداتين البرمجيتين المتوفرتين (BABEL-tool, BPMN2BPEL plug in) من حيث دعمه لمختلف مكونات التمثيل الرمزي لإجرائيات العمل وقابلية قراءة رماز اللغة التنفيذية الناتج.

6.2. الآفاق المستقبلية

قمنا ضمن الفصل السابق بتقييم نموذج التحويل الذي قمنا ببنائه ضمن الفصل الثالث وذكرنا نقاط ضعفه وقوته، وناقشنا الشروط المفروضة على نموذج الدخل وجدنا أنه من الممكن أن نتراخى ببعض شروط البناء الجيد لمخطط إجرائية العمل المستخدم كدخل لنموذج التحويل، وهذا يفترض بعض التعديلات البسيطة على التنجيز البرمجي لمعالجة النماذج التي لا تحقق شروط البناء الجيد بهدف تحويلها إلى نماذج تحقق هذه الشروط.

كما وجدنا عدم دعم النموذج المقترح للحلقات غير المهيكلة والتي يفترض مراعاتها مستقبلاً، حيث وجدنا دعم هذه الحلقات ضمن الأداة (BABEL). كذلك فإن التنجيز البرمجي يحتاج إلى اختبار على نماذج تمثيل رمزي متعددة ومتنوعة ومعالجة بعض الأخطاء التي ممكن أن تظهر مع النماذج المعقدة، وبعد ذلك يمكن تطوير هذا التنجيز ليشكل إضافة على Eclipse تكون سهلة الاستخدام.

المراجع

1- Object Management Group, Business Process Modeling Notation (BPMN). <http://www.bpmn.org/>

2- Web Services Business Process Execution Language Version 2.0, OASIS Standard, 11 April 2007 available via < **http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html**>

3- R. Khalaf, N. Mukhi, S. Weerawarana **Service–Oriented Composition in BPEL4WS**, 2003

4- J. Recker and J. Mendling. **On the translation between BPMN and BPEL: Conceptual mismatch between process modeling languages**. Proceedings 18th International Conference on AdvancedInformation Systems Engineering, pages 521–532, 2006.

5- J. Mendling, K.B. Lassen, and U. Zdun. **On the transformation of control flow between block-oriented and graph-oriented process modeling languages**. Int. J. Business Process Integration and Management, 2006.

6- C. Ouyang, W.M.P van der Aalst, M. Dumas, and A.H.M. ter Hofstede. **From BPMN process models to BPEL web services**. Proceedings of the 4th International Conference on Web Services, IEEE Computer Society: pages 285–292, 2006.

7- C. Ouyang, W.M.P. van der Aalst, M. Dumas, and A.H.M. ter Hofstede. **From business process models to process-oriented software systems**: The BPMN to BPEL way. ACM Transactions on Software Engineering and Methodology, 2008.

8- C. Ouyang, M. Dumas, A.H.M. ter Hofstede, and W.M.P. van der Aalst. **Pattern-based translation of BPMN process models to BPEL web services**. International Journal of Web Services Research, 5(1):42–62, 2008.

9- S.A. White. **Using BPMN to model a BPEL process**. IBM white paper, 2005.

10- Zhan Shi, Xiaoqin Zeng, Song Huang, Hui Li, Bing Hu, XiaoYu Lei, Yi Wang. **Transformation between BPMN and BPEL based on graph grammar**, 2015.

11- J. Vanhatalo, H. V¨olzer, and J. Koehler. **The refined process structure tree**. BPM, pages 100–115, 2008.

12- **bpmn2bpel toll,** < https://code.google.com/p/bpmn2bpel>

13- R.M. Dijkman, M. Dumas, and C. Ouyang. **Semantics and analysis of business process models in bpmn**. Information and Software Technology, 50(12):1281–1294, 2008.

14- W.M.P. van der Aalst and K.B. Lassen. **Translating unstructured workflow process to readable BPEL**: Theory and implementation. Information and Software Technology, Volume 50 , Issue 3:131–159, 2008.

15- L.Garc´ıa-Ba˜nuelos. **Pattern identification and classification in the translation from bpmn to bpel**. In OTM Conferences (1), pages 436–444, 2008.

16- R. Eshuis and P.W.P.J. Grefen. **Composing services into structured processes**. International Journal of Cooperative Information Systems, to be appear, 2009.

17- F. van Breugel and M. Koshkina. Dead-path-elimination in BPEL4WS. Fifth International Conference on Application of Concurrency to System Design, 2005.

18- R. Eshuis. **Statecharting petri nets**. BETA Working Paper Series WP 153, 2005.

19- A.V. Aho, R. Sethi, and J.D. Ullman. Compilers: Principles, Techniques, and Tools. Addison Wesley, 1986.

20- R. Eshuis and A. Kumar. **Converting Unstructured into Semi-Structured Process Models**, 2015.

21- B. Kiepuszewski, A. ter Hofstede, C. Bussler, **On structured workflow modelling**. CAiSE, Springer, 2000, pp. 431–445.

22- R. Liu, A. Kumar, **An analysis and taxonomy of unstructured workflows**. 3rd Conference on Business Process Management (BPM 2005), pp. 268–284.

23- J. Roa, O. Chiotti, P. Villarreal, **Verification of Structured Processes**: **A Method Based on an Unsoundness Profile**. 14th Argentine Symposium on Software Engineering, ASSE 2013.