1. Introducción

El propósito de este trabajo es identificar y documentar las reglas de negocio presentes en los programas COBOL y sus componentes asociados, con el objetivo de facilitar su comprensión y trazabilidad.

El análisis se centra en tres programas principales (COBAP01, CICBP02 y UTLVAL01), los copybooks ACCREC y PARAMS, y el JCL que controla la secuencia de ejecución.

La metodología aplicada se basa en técnicas de análisis estático: generación de Árboles de Sintaxis Abstracta (AST) y construcción de un grafo semántico que refleja dependencias, relaciones y trazabilidad entre variables, estructuras y reglas de negocio. Este enfoque permite separar la lógica de negocio real de los elementos técnicos accesorios, ofreciendo una visión clara y auditable del comportamiento del sistema.

2. Inventario de componentes

Programas:

COBAP01: cálculo de intereses y control de bloqueo de cuentas.

CICBP02: autorización de transacciones según estado, riesgo y balance.

UTLVAL01: validación de comisiones en modo test.

Copybooks:

ACCREC: define la estructura del registro cuenta.

PARAMS: contiene parámetros de ejecución y de negocio.

JCL JOB1: controla la secuencia de ejecución.

3. Reglas de negocio extraídas

COBAP01: procesa el archivo secuencial ACC.DAT con registros de cuentas (copy ACCREC) aplicando dos reglas de negocio principales: cálculo de intereses sobre saldos de ahorro y bloqueo de cuentas con sobregiros no permitidos.

El cálculo de interés se aplica a cuentas con estado abierta, de tipo ahorro, sobre el campo ACC-BALANCE y en modo productivo la tasa de interés es positiva.

Si el programa se corre en modo test la tasa se fuerza a 0 , con lo que la operación se ejecuta pero no genera incremento sobre ACC-BALANCE.

Respecto el bloqueo, si el parámetro PRM-ALLOW-NEG está desactivado y el balance de la cuenta es menor que el negativo del límite de sobregiro (–ACC-OD-LIMIT), la cuenta se marca como bloqueada.

Esta regla se aplica tanto en modo productivo como en modo test.

Se distingue entre lógica productiva y lógica de prueba, se mantiene activa la regla de bloqueo pero sin impacto en balance.

CICBP02: decide la autorización de operaciones sobre cuentas, aplicando criterios de negocio que combinan estado, riesgo y balance.

La lógica principal establece que las cuentas bloqueadas, con nivel de riesgo elevado (> 5) o en caso que la suma del saldo de la cuenta + el límite de sobregiro permitido es 0 o negativa, la autorización se rechaza.

El programa actúa como filtro de seguridad, garantizando que las operaciones cumplan con las condiciones de negocio definidas.

UTLVAL01: es un programa validador.
Si se ejecuta en modo test, incrementa el contador WS-CNT y asigna 0 al campo ACC-FEE-COUNT lo que neutraliza la tabla ACC-FEE del registro de la cuenta ACC-RECORD. Se simula un escenario en donde las reglas de comisiones no se validan correctamente, ya que el sistema actúa como si no existieran, lo cual no refleja el comportamiento real del negocio.
En modo productivo solo incrementa el contador WS-CNT registrando que se ejecutó una verificación técnica, en este caso funciona como un módulo auxiliar de control sin alterar balances ni estados de cuenta.

Copys: definición de estructura de cuenta y parámetros.
Los copybooks ACCREC y PARAMS constituyen la base de datos lógica sobre la que operan los programas principales.
ACCREC define la estructura de la cuenta, incluyendo número, tipo, estado, balance, límite de sobregiro, riesgo y comisiones (grupo AC-FEE que puede ocurrir de 0 a 5 veces dependiendo del valor de ACC-FEE-COUNT).
PARAMS: parámetros de control que afectan al comportamiento de los programas.
TEST-MODE, indica si el programa se ejecuta en modo prueba
DATE-CUTOFF, parámetro de corte o fecha límite
ALLOW-NEGATIVE, controla si se permiten saldos negativos
Si está en 'Y', el sistema permite que las cuentas tengan balance negativo (sobregiro).
Si está en 'N', las cuentas con saldo inferior al negativo del límite de sobregiro (–ACC-OD-LIMIT).
Estos componentes no contienen lógica de negocio por sí mismos, pero son esenciales para los programas COBAP01, CICBP02 y UTLVAL01.

JCL: es el componente que articula la ejecución y controla las condiciones de entrada y salida de cada etapa.
No aporta reglas de negocio en sí mismo, pero sí condiciona su aplicación, convirtiéndose en un punto crítico para la trazabilidad y la confiabilidad del proceso.
Su función principal es garantizar que los programas se ejecuten en el orden correcto y con los parámetros adecuados, pero también introduce lógica de control que puede modificar el comportamiento global.
La secuencia esperada sería COBAP01 → CICBP02 → UTLVAL01
Pero si COBAP01 finaliza OK (RC=0) --> el flujo se corta y CICBP02 no se ejecuta.
Primero corre COBAP01, que procesa el archivo de cuentas, calcula intereses y bloquea cuentas según reglas.
Luego debería correr CICBP02, que llama dinámicamente a UTLVAL01.
Pero la condición COND=(0,NE) impide ejecutar CICBP02 si el programa COBAP01 finaliza OK, hace que CICBP02 sólo se ejecute si el programa COBAP01 falla.
Esto rompe la cadena de programas normal y es un error intencional, lo normal sería que COBAP01 finalice OK y luego CICBP02 continúe la cadena.
En condiciones normales se esperaría que la autorización CICBP02 corra después del cálculo COBAP01, pero CICBP02 no se ejecuta, esta condición capciosa rompe la cadena de negocio.

4. Errores y hallazgos técnicos
Se identificaron varios aspectos técnicos que afectan la trazabilidad y la confiabilidad del flujo de ejecución. Estos hallazgos no constituyen reglas de negocio en sí mismos, pero condicionan su aplicación y deben ser documentados.

Condición de fin de archivo en COBAP01 (3000-END-OF-FILE) → error de diseño.
El programa COBAP01 utiliza una condición de fin de archivo que no está correctamente diseñada.
La rutina 3000-END-OF-FILE fuerza la terminación del proceso sin un cierre ordenado de recursos ni un control explícito de estado.
Esto constituye un error de diseño, ya que interrumpe la ejecución de manera abrupta y dificulta la trazabilidad del flujo.
Una solución sería implementar una rutina de lectura con controles explícitos de OPEN y CLOSE sobre ACC-FILE, manejando además el fin de archivo mediante File Status (FS) para garantizar un cierre ordenado, trazable y que permita diferenciar entre un fin de archivo normal y un error de I/O.

Condición en JCL (COND=(0,NE)) → flujo truncado, error intencional.
El JCL que controla la secuencia de ejecución introduce una condición capciosa: COND=(0,NE).
Esta instrucción provoca que CICBP02 no se ejecute si COBAP01 finaliza correctamente (RC=0).
En consecuencia, el flujo normal queda truncado y CICBP02 solo corre si COBAP01 falla.
Se trata de un error intencional que rompe la cadena de negocio esperada, ya que la autorización debería ejecutarse después del cálculo de intereses y bloqueos.
Este hallazgo es crítico porque altera la lógica global del proceso. La solución sería eliminar o modificar la condición para garantizar que CICBP02 corra siempre después de COBAP01, manteniendo la cadena de negocio intacta.

Flag sin uso (WS-UNUSED-FLAG).
En el programa COBAP01 se detectó un campo declarado en la Working-Storage Section (WS-UNUSED-FLAG) que nunca es referenciado en el código.
Las variables sin uso dificultan la comprensión y afectan la limpieza y mantenimiento del código.
Se sugiere eliminar el flag o documentarlo como reservado mediante comentario en la definición.

Logging simulado (WS-TMP-BYTE).
El programa COBAP01 contiene una variable declarada (WS-TMP-BYTE) utilizada como mecanismo de logging simulado.
El programa pasa por el bloque IF, evalúa la condición y continúa sin ejecutar ninguna acción.
Este hallazgo muestra una intención de instrumentación técnica que nunca se concretó.
La condición IF nunca se cumple en condiciones normales porque el valor numérico 0 se representa como X'30' (ASCII) o X'F0' (EBCDIC), no como X'00', por lo que no aporta trazabilidad real.
La solución sería implementar un mecanismo de logging efectivo (archivo o consola) o eliminar el bloque.

5. AST y Grafos Semánticos:
En este punto se incluyen fragmentos correspondientes a los AST generados automáticamente por el pipeline de análisis estático con las principales variables, párrafos y sentencias de los programas, copys y pasos del job.
Se presenta también una representación resumida del AST en formato árbol.

La representación (AST técnico + árbol textual) facilita tanto la lectura técnica como la auditoría ejecutiva.

COBAP01


```
{
  "fileName": "COBAP01.cbl",
  "dataVariables": ["WS-INT-RATE", "WS-DATE", "ACC-RECORD"],
  "procedureParagraphs": ["1000-INIT", "2000-PROCESS", "9000-FINALIZE"],
  "procedureStatements": ["ReadStatementImpl", "IfStatementImpl", "CloseStatementImpl"]
}
```


```
PROGRAM COBAP01
 └─ PROCEDURE DIVISION
    ├─ 1000-INIT → inicializa fecha
    ├─ 2000-PROCESS → intereses y bloqueo
    └─ 9000-FINALIZE → cierre de archivo
```

COBAP01 procesa el archivo de cuentas, calcula intereses en cuentas de ahorro abiertas y bloquea cuentas con balance negativo si no se permiten sobregiros.
El AST de COBAP01 permite identificar reglas de cálculo, de bloqueo y dependencias.
Regla de intereses: si la cuenta está abierta y es de ahorro, calcula intereses y lo suma al balance.
Regla de bloqueo: si el balance es menor al negativo del límite de sobregiro (–ACC-OD-LIMIT) y no se permiten negativos, la cuenta se bloquea.
Dependencias: estas reglas de negocio se apoyan en los niveles 88 definidos en el copy ACCREC (estado, tipo, balance, límite):

ACC-STATUS	88 ACC-OPEN	VALUE 'O'
ACC-TYPE	88 ACC-TYPE-SAVINGS	VALUE 'S'
(NOT PRM-ALLOW-NEG)	88 PRM-ALLOW-NEG	VALUE 'Y'
(ACC-BALANCE < -ACC-OD-LIMIT)	(balance < límite de sobregiro)	

Se detecta la variable WS-UNUSED-FLAG que no se utiliza y un control de fin de archivo redundante (3000-END-OF-FILE).

CICBP02


```
{
  "fileName": "CICBP02.cbl",
  "dataVariables": ["ACC-RECORD", "PRM-FLAGS", "WS-AUTH-DECISION"],
  "procedureParagraphs": ["0000-MAIN"],
  "procedureStatements": ["MoveStatementImpl", "EvaluateStatementImpl", "CallStatementImpl"]
}
```


```
PROGRAM CICBP02
 └─ 0000-MAIN
    ├─ MOVE LK-ACC TO ACC-NUMBER
    ├─ EVALUATE → estado, riesgo, balance
    ├─ CALL WS-MOD-NAME USING ACC-RECORD
    └─ GOBACK
```

CICBP02 autoriza operaciones según estado, riesgo y balance.
El AST permite identificar las siguientes reglas de negocio y dependencias:
Rechaza cuentas bloqueadas o con riesgo alto
Regla de rechazo por estado: Si cuenta bloqueada → se rechaza la operación.
Regla de rechazo por riesgo: Si riesgo > 5 → se rechaza la operación.
Regla de autorización:
Si balance + OD-LIMIT (límite de sobregiro) > 0 → autorización, de lo contrario → rechazo.
Dependencias: estas reglas se apoyan en los campos definidos en el copy ACCREC:
ACC-STATUS 88 ACC-BLOCKED VALUE 'B'
ACC-RISK-RATING > 5
ACC-BALANCE + ACC-OD-LIMIT > 0
Se invoca al programa validador UTLVAL01 usando el registro ACC-RECORD y los parámetros de PRM-FLAGS. Esta llamada dinámica puede alterar ACC-FEES, introduciendo una dependencia externa. WS-MOD-NAME hardcodeado a 'UTLVAL01' (podría cambiarse).

UTLVAL01


```
{
  "fileName": "UTLVAL01.cbl",
  "dataVariables": ["WS-CNT", "ACC-RECORD", "PRM-FLAGS"],
  "procedureParagraphs": ["100-VALIDATE"],
  "procedureStatements": ["AddStatementImpl", "IfStatementImpl"]
}
```


```
PROGRAM UTLVAL01
 └─ 100-VALIDATE
    ├─ ADD 1 TO WS-CNT
    ├─ IF PRM-TEST-YES → reset fees
    └─ GOBACK
```

UTLVAL01 es un validador técnico.
Es invocado dinámicamente por CICBP02, lo que introduce una dependencia externa y condiciona la lógica de autorización.
Aunque parece técnico impacta en el negocio porque elimina comisiones que podrían ser usadas en cálculos posteriores.
El AST de UTLVAL01 permite identificar las siguientes reglas de negocio y dependencias:
Regla de validación: cada vez que se invoca el programa, se incrementa el contador de validaciones WS-CNT.
Regla de modo test: si el parámetro PRM-TEST-YES está activo, se fuerza ACC-FEE-COUNT = 0, eliminando las comisiones asociadas a la cuenta.
PRM-TEST-MODE 88 PRM-TEST-YES VALUE 'Y'
Dependencias: el programa recibe como parámetros ACC-RECORD y PRM-FLAGS, que lo conectan directamente con los copys ACCREC y PARAMS.

ACCREC


```
{
  "fileName": "ACCREC-STUB.cbl",
  "dataVariables": ["ACC-NUMBER", "ACC-TYPE", "ACC-STATUS", "ACC-BALANCE", "ACC-OD-LIMIT"]
}
```

COPY ACCREC
├─ ACC-NUMBER
├─ ACC-TYPE (CHECKING, SAVINGS)
├─ ACC-STATUS (OPEN, CLOSED, BLOCKED)
├─ ACC-BALANCE, ACC-OD-LIMIT
└─ ACC-FEES (OCCURS DEPENDING ON ACC-FEE-COUNT)

El copy ACCREC define la estructura del registro cuenta y es base para las reglas de los programas COBAP01, CICBP02 y UTLVAL01.
número (ACC-NUMBER)
tipo (ACC-TYPE)
estado (ACC-STATUS)
balance (ACC-BALANCE)
límite (ACC-OD-LIMIT)
riesgo (ACC-RISK-RATING)
comisiones (ACC-FEES).
Estado de la cuenta: niveles 88 (ACC-OPEN, ACC-CLOSED, ACC-BLOCKED).
Tipo de cuenta: niveles 88 (ACC-TYPE-CHECKING, ACC-TYPE-SAVINGS).
Balance (ACC-BALANCE) y límite de sobregiro (ACC-OD-LIMIT) utilizados en COBAP01 y CICBP02 para calcular intereses, bloquear cuentas (comparando contra el negativo del límite) y autorizar operaciones.
Riesgo (ACC-RISK-RATING) utilizado en CICBP02 para rechazar operaciones con riesgo alto.
ACC-FEES es un array dinámico o tabla interna de comisiones por cuenta de hasta 5 registros, el número real de ocurrencias depende del valor de la variable ACC-FEE-COUNT
Esta variable es manipulada en UTLVAL01, en modo test se fuerza ACC-FEE-COUNT=0.

PARAMS

```
{
  "fileName": "PARAMS-STUB.cbl",
  "dataVariables": ["PRM-TEST-MODE", "PRM-DATE-CUTOFF", "PRM-ALLOW-NEGATIVE"]
}
```


COPY PARAMS
├─ PRM-TEST-MODE (TEST-YES)
├─ PRM-DATE-CUTOFF
└─ PRM-ALLOW-NEGATIVE (ALLOW-NEG)

PARAMS define los parámetros de ejecución: modo test, fecha de corte y permitir negativos.
Condiciona reglas en los programas COBAP01 y UTLVAL01.
Modo test (PRM-TEST-YES): utilizado en UTLVAL01 para eliminar comisiones durante pruebas.
Fecha de corte (PRM-DATE-CUTOFF): parámetro de control temporal.
Permitir negativos (PRM-ALLOW-NEG): condiciona la regla de bloqueo en COBAP01.
Si está activo, se permite balance negativo, si está en 'N', la cuenta se bloquea cuando el balance es menor que el negativo del límite de sobregiro (–ACC-OD-LIMIT).

JOB COBAP01


```
{
  "jobName": "COBAP01",
  "steps": [
    {"name": "STEP1", "program": "COBAP01"},
    {"name": "CICBSTEP", "program": "CICBP02", "condition": "(0,NE)"}
  ]
}
```


JCL JOB COBAP01

```
 ├─ STEP1 → EXEC PGM=COBAP01
 └─ CICBSTEP → EXEC PGM=CICBP02 COND=(0,NE)
```

El JCL COBAP01 permite identificar las siguientes dependencias y condiciones de ejecución:
STEP1: ejecuta el programa COBAP01 y genera salida en SYSOUT.
CICBSTEP: ejecuta el programa CICBP02 solo si la ejecución del programa COBAP01 termina con código distinto de cero (COND=(0,NE)).
Esta condición implica que si COBAP01 finaliza correctamente, CICBP02 no se ejecuta, lo que puede cortar el flujo de negocio esperado.

6. Conclusiones y recomendaciones

Reglas de negocio identificadas

El análisis estático permitió documentar de manera clara las reglas de negocio críticas en los programas COBOL:
. Cálculo de intereses en cuentas de ahorro abiertas.
. Bloqueo de cuentas cuando el balance es menor que el negativo del límite de sobregiro (–ACC-OD-LIMIT) y no se permiten sobregiros.
. Autorización de operaciones condicionada por estado de cuenta, rating de riesgo y la suma de balance más límite de sobregiro disponible.
. Eliminación de comisiones en modo test.
Estas reglas fueron trazadas desde los ASTs hasta el grafo semántico, asegurando transparencia y auditabilidad.

Riesgos por errores de diseño

Durante el análisis se identificaron riesgos técnicos que afectan la lógica del flujo de ejecución
. Control de EOF en COBAP01, que introduce complejidad innecesaria y carece de cierre ordenado.
. Condición JCL (COND=(0,NE)) que impide la ejecución de CICBP02 si COBAP01 finaliza correctamente, cortando el flujo de negocio esperado.
. Variables sin uso y bloques de logging simulado que generan ruido técnico y debilitan la trazabilidad.

Importancia de separar reglas reales de elementos técnicos irrelevantes

El proceso evidenció la necesidad de distinguir entre:
. Reglas de negocio reales, que impactan directamente en clientes y operaciones.
. Elementos técnicos irrelevantes, como variables sin uso, controles redundantes y sentencias de salida.

Esta separación es clave para que el documento sea consumible tanto por auditores como por modelos de lenguaje.

Recomendaciones

. Rediseñar el control de EOF en COBAP01, incorporando OPEN/CLOSE explícito y manejo de File Status (FS) para diferenciar entre fin de archivo normal y error de I/O.

. Revisar y corregir la condición en JCL para asegurar que CICBP02 se ejecute siempre después de COBAP01, manteniendo la cadena de negocio intacta.

. Eliminar o documentar como reservadas las variables sin uso, evitando confusión en futuros mantenimientos.

. Sustituir el logging simulado por un mecanismo real de registro (archivo o consola) o eliminarlo si no se requiere trazabilidad.

. Mantener la documentación dual (AST técnico + árbol textual) para garantizar trazabilidad y comprensión tanto para auditores como para equipos de desarrollo.

En conclusión, el análisis permitió separar reglas de negocio críticas de componentes técnicos no significativos y documentar dependencias con transparencia.
Los hallazgos técnicos identificados requieren corrección inmediata para garantizar confiabilidad y trazabilidad en el flujo de ejecución.
La aplicación de las soluciones propuestas asegura que el sistema sea más limpio, auditable y alineado con las reglas de negocio reales, fortaleciendo su valor tanto técnico como ejecutivo.