

# Program sanity checking in a CA Plex environment

11A

Morten Knudsen

KODA

Co-branded Logo Footprint  
Aligned LEFT ON COVER ONLY  
Must Fit Within This Space

**ADC AUSTIN**

**MKS**

**WEBSYDIAN™**

**cm** FIRST

**desynit**  
**ca**

## > Contents

Speaker and Background

Sanity Checking

Basic Plex Implementation

Tracking Errors

Extended Logging

Practical Experience

Co-branded Logo Footprint  
Aligned Right Edge  
Must Fit Within This Space

---

# Speaker and Background

About Speaker

About KODA

Using Plex in KODA

# Morten Knudsen

---

- > MS. Computer science
- > 1996 Zurich Insurance (2E and Plex)
  - Development, Conversion, Methodology, Documentation
- > 1998 Soft Design (Plex/Websydian)
  - Consulting, Education, Websydian product development, Documentation
- > 2003 Independent consultant
  - Development
- > 2003 KODA (Plex/Websydian)
  - Project management, Design architecture, Development, Strategy

# What Is KODA?

- > *"KODA is a non-profit collective rights management society that administers Danish and international copyrights for music creators and publishers, when their music is performed in public."*
- > *"KODA's two main tasks are to ensure that the music creators get paid when their music is played in public - and to ensure that the music users are able to clear the rights to the music they wish to use."*
- > *"KODA represents approximately 33,000 Danish composers, songwriters and music publishers. Through reciprocal contracts with rights societies in more than 115 countries, we offer the entire world repertoire of music in one single agreement."*

# Five Good Reasons to Choose KODA

- > Very low administration costs
- > Creator of new business models for music use
- > Prime mover on online administration
- > Efficiency and innovation
- > Collaboration nationally and internationally

Can only be achieved if supported by  
powerful and efficient IT systems

# Plex in KODA

---

- > 8 Plex models/applications
- > 5 full-time Plex developers (+ 2 Navision developers)
- > AS/400 platform – database and web server
- > 5250 and web
- > Provider of IT systems for Norwegian TONO and Swedish STIM
- > Complete migration from 2E applications

# Sanity Checking

Acknowledgement to Christian Ernstsén

Different Ways to Detect Application Errors

What Is Sanity Checking – and Why Do It?

# Acknowledgement to Christian Ernstsén

---

- > Sanity checking in KODA implemented from idea first expressed by Christian Ernstsén (former employee of Soft Design and KODA)

# Different Ways to Detect Application Errors

- > Developers test of own programs
  - Detected and corrected by developers
- > User test as part of project
  - Detected by user representatives in development team
- > Run-time errors
  - Detected by users doing their work
- > Background/internal errors
  - Detected by system administrators supervising the systems

Sanity Checking supports all phases of  
program error finding

# Symptoms and causes

- > Errors found are detected by their symptoms
- > The symptom of an error does not always indicate the cause of the error

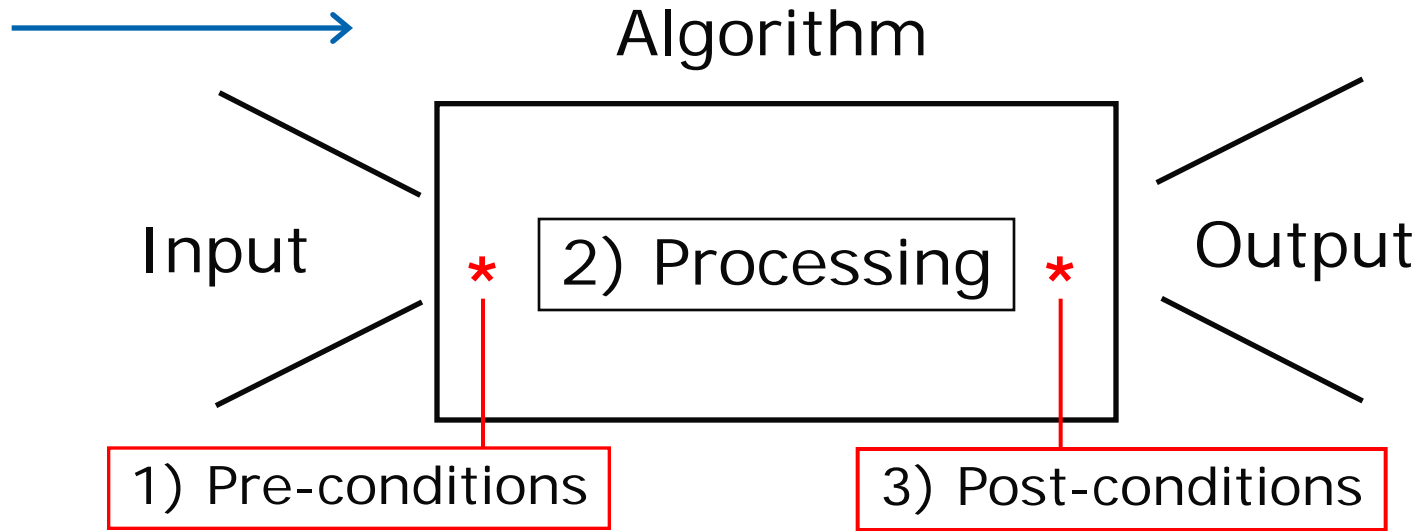
Sanity Checking supports detecting the cause of errors

Sanity Checking supports early detection of errors

# Sanity Checking/Testing – Definition

- > *“A sanity test is a very brief run-through of the functionality of a computer program, system, calculation, or other analysis, to assure that the system or methodology works as expected” [wikipedia].*
- > Our interpretation of *sanity checking*: Validate that everything is as it is supposed to be – and report back if this is not the case...

# Pre- and Post Conditions of Algorithms



- > *Sanity checking* could be to analyze pre- and post conditions for every implemented algorithm (function) and check that those were fulfilled
- > We have chosen a simpler implementation...

# Basic Plex Implementation

Using *\*Returning status* to detect abnormal program state

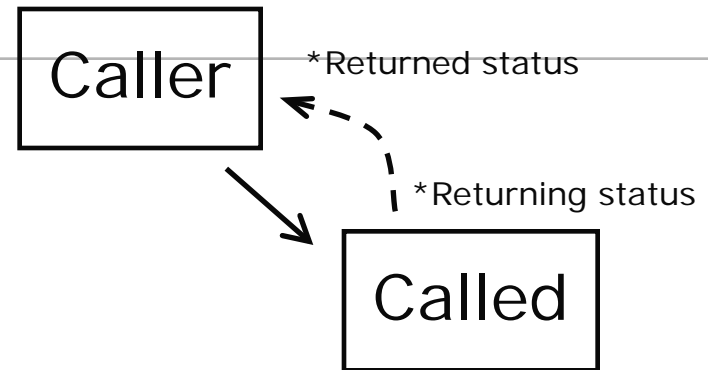
Not all abnormal states are abnormal

Logging unexpected behavior

Basic pattern used for sanity checking

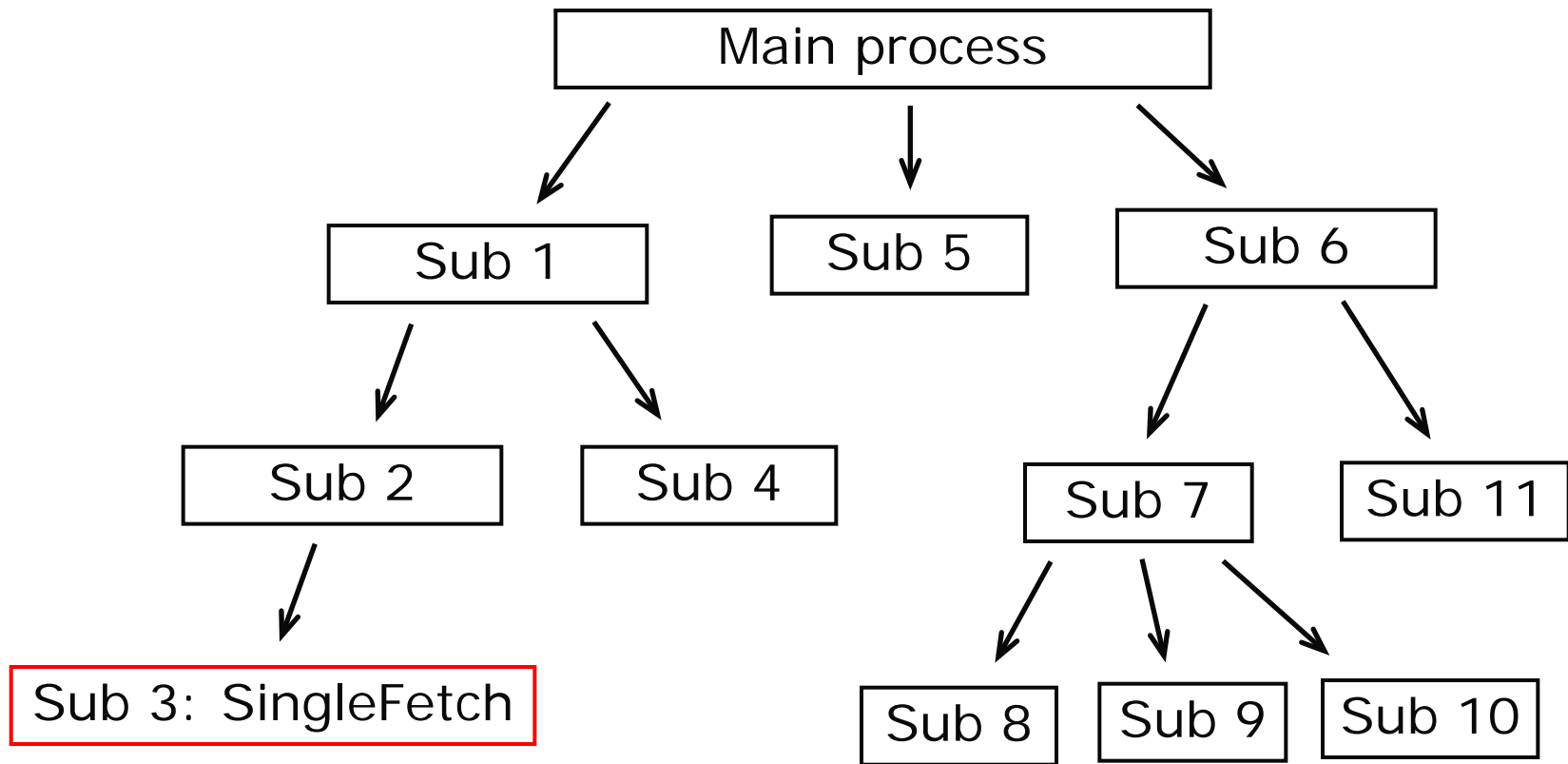
# Plex *\*Returning status* / *\*Returned status* – an unused feature

- > *\*Returning status* from called function received as *\*Returned status* by calling function
- > Checking of abnormal *\*Returning status* implemented in some client programs calling server programs
- > Checking is done explicitly



```
Action Diagram: [(Read-Only)]Detail
returned s
Sub Load Detail
+++Define Field: FIELDS/+Subroutine
  Edit Point Start load Detail
+If Field: FIELDS/+Subroutine
  Load the panel data from the database
  Call UIBASIC/UIBasic.Detail.SingleFetch
  Edit Point Post fetch processing
  Case
    When Environment<*Returned status> == <*Returned status.*Instance not found>
      +++Set Value Field: OBJECTS/*Message, Message: FIELDS/NoData
      +For Defined Value Field: OBJECTS/*Message
      Format Message Message: FIELDS/NoData, Environment<*Message text>
      The data could not be read from the database - no row exists for the supplied keys.
      Edit Point Fetch Detail failed - no data.
      Go Sub Send message
    When Environment<*Returned status> == <*Returned status.*Instance locked>
    When Environment<*Returned status> IS <State: OBJECTS/*Returned status.*Abnormal>
      +++Set Value Field: OBJECTS/*Message, Message: FIELDS/UnknownError
      +For Defined Value Field: OBJECTS/*Message
      Format Message Message: FIELDS/UnknownError, Environment<*Message text>
      Edit Point Fetch Detail failed - unknown error.
      Go Sub Send message
    When Environment<*Call status> IS <State: OBJECTS/*Call status.*Abnormal>
    Otherwise
```

# Example – Error Not Detected



> “Instance not found” error by Sub 3 is not detected

# Abnormal *\*Returning status* – not always an error

## > *SingleFetch* and *UpdateRow*

- *\*Instance not found*: Alternative processing may be defined if record not found

## > *BlockFetch*

- *\*End of view, \*Instance not found*: Tell calling client function that no more records in defined set

## > *CheckRow*

- *\*Instance not found*: Called by *InsertRow* function to check if record with same primary key already exists

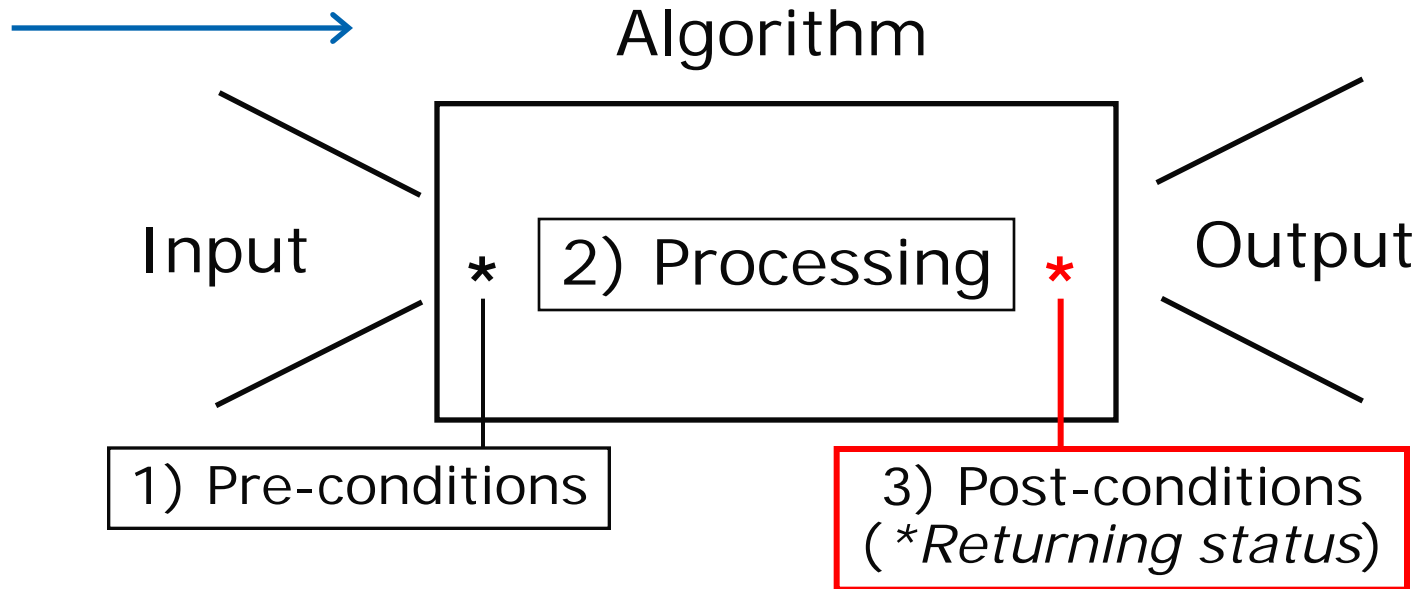
## > *InsertRow*

- *\*Instance exists*: If record with same primary key already exists

# Using Plex *\*Returning status* to detect abnormal program state

- > Log abnormal *\*Returning status* for all server programs
  - Log called program and input parameters...
- > De-select standard cases (defined in abstract layer)
  - *CheckRow: \*Instance not found*
  - *BlockFetch: \*End of view, \*Instance not found*
- > Further de-selection must be possible in specified cases
  - De-selection by options...

# Using Plex *\*Returning status* to detect abnormal program state

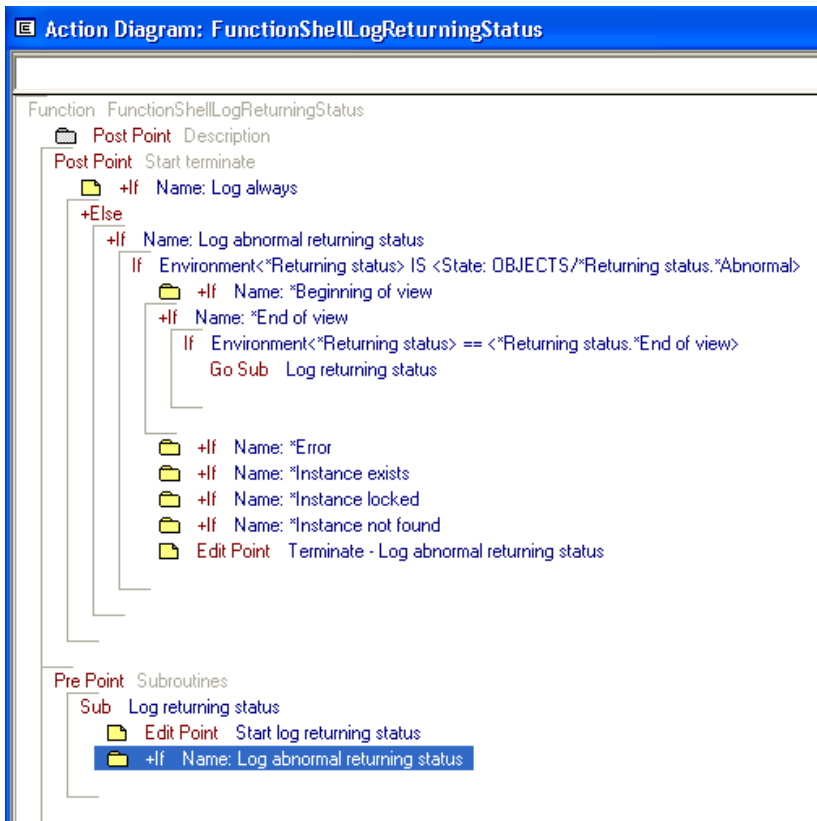


- > Validation of *\*Returning status* post condition
- > Not a complete sanity check

Log when *\*Returning status* is not as expected

# Basic pattern used abnormal *\*Returning status* logging

- > *FunctionShellLogReturningStatus* in inheritance path of all server functions

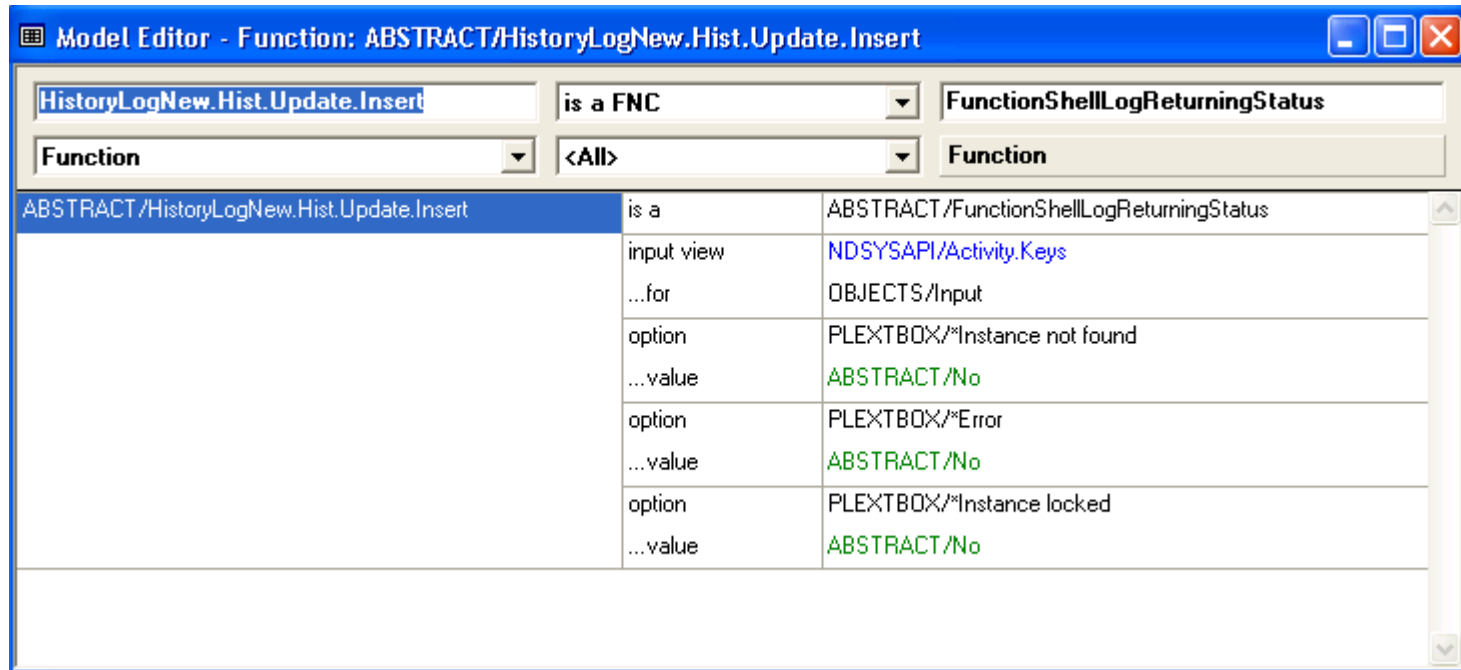


- > Go sub *Log returning status* sub on abnormal *\*Returning status*
- > Each *\*Returning status* value may be switched of by option
- > All logging may be switched of by *Log abnormal returning status* name option

Simple basic implementation!

# Switch off \*Returning status Logging by options

> Example of (abstract) function with deselected logging



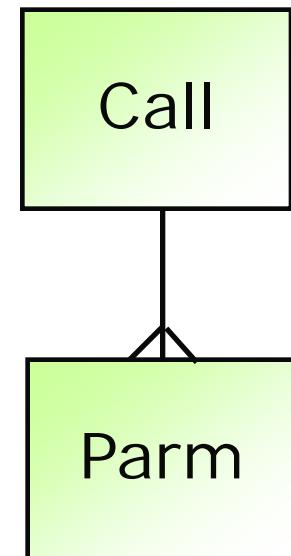
# Log function call and parameters in database tables

## > *Call* entity

- Implementation name, Function name, Returning status
- User, Date, Time, Job number, Job name
- Calling program 1-9

## > *Parm* entity

- Owned by *Call* entity
- Known by Sub sequence
- Input variable
- Field implementation name
- Field value



# Meta code to log function header and .input parameters

```
Sub Log returning status
  Pre Point
  Edit Point Start log returning status
  Post Point
  +If Name: PLEXTBOX/Log abnormal returning status
    +++Define Field: OBJECTS/Meta variable
    Name Function: FunctionShellLogReturningStatus [FunctionShellLogReturningStatus], Environment<*Object>
    Set Environment<*View status> = Environment<*Returning status>
    +++Define Field: OBJECTS/*Boolean
    +For Each Property Target FNC impl name NME
    +If Field: OBJECTS/*Boolean
      Pre Point
      Edit Point Log called function header
    Post Point
    Call NDSYSAPI/Log.Server.Log function header
    Set Local<&Log Sub sequence> = <&Log.Sub sequence.*One>
    +++Define Field: FIELDS/+Variable
    +For Each Variable .Input
      +++Set Value To Current Field: FIELDS/+Variable
      ++Name Defined Field: FIELDS/+Variable, Environment<*Message text>
      +For Each Field
        +++Define Field: OBJECTS/*Boolean
        +For Each Field Component Do not log structured fields
          +If Field: OBJECTS/*Boolean
            +Case Get field impl name
              +When ATR impl name NME
              +When FLD impl name NME
              +Otherwise
            ++Cast To Environment<*Object>
            Pre Point
            Edit Point Log function input parameter
          Post Point
          Go Sub Log function input parameter
```

Log function header

Log input parameter

Simple basic implementation!

# Tracking Errors

Browsing and Administrating Log Data

Retrieving Call Program Stack

Email Notification

*Log always* Option for Debugging

# Browsing and Administrating Log Data – By 5250 interface

```

Session B - [27 x 132]
File Edit View Communication Actions Window Help
NSRetLog Display EDBMKN List abnormal function calls 5-09-09 22:49:14

Seq. _____ Program name _____ Returning status ___ Job number _____ Vis parm. Y

1=Filter on pgm. 4=Clear 5=Details 7=Clear all
Seq# Program Full function name Sts Date Time User Develo Field value string
-----
_ 50640421 TBEXECMD AS400 utilities.Exec (S10 S1124091640 S1124099683) INT 04-09-09 18:43:48 KODJVI EDBFKR TBEXECMD|202|202|252|TB
_ 50640395 NM4iF Match.Run.Update.Set state INF 04-09-09 16:38:30 KODJTH EDBFKR N|0|F
_ 50640389 NR1g1F Report.Period.Update.Increment title count\id date INF 04-09-09 16:12:29 KODJTH EDBFKR N|0|0|1
_ 50640388 NR5lyF Program.Program report.Fetch.SF Ch\date\time INF 04-09-09 16:12:29 KODJTH EDBFKR 0
_ 50640387 NR6saF Program.Program report.Fetch.SF Cue\CR\TE\RS\ND\RR INF 04-09-09 16:12:28 KODJTH EDBFKR 0
_ 50640386 NRf1F Report.Report.Fetch.SingleFetchCopyFields INF 04-09-09 16:12:28 KODJTH EDBFLA 0
_ 50640385 NR1fcF Match.Manually matched title.Update.UpdateCopyFiel INF 04-09-09 16:12:28 KODJTH EDBHOC 29393042|B|DR|N
_ 50640377 NM4iF Match.Run.Update.Set state INF 04-09-09 15:37:36 KODJTH EDBFKR N|0|F
_ 50640376 NM4iF Match.Run.Update.Set state INF 04-09-09 15:37:26 KODJTH EDBFKR N|0|F
_ 50640375 NRpsF List.Report from web.By web user\party\date D.Bloc ERR 04-09-09 15:37:15 3513 EDBHOC 7 3513|1|Y|0
_ 50640373 NR3qbF MShop.Match update IEX 04-09-09 15:34:24 KODJTH EDBFKR M|M|493612000015|372327
_ 50640372 NR3qbF MShop.Match update IEX 04-09-09 15:33:11 KODJTH EDBFKR M|M|493612000015|372584
_ 50640371 NR3qbF MShop.Match update IEX 04-09-09 15:31:23 KODJTH EDBFKR M|M|493612000015|372527
_ 50640365 NR1fcF Match.Manually matched title.Update.UpdateCopyFiel INF 04-09-09 15:07:53 KODJTH EDBHOC 29357389|B|DR|N
_ 50640364 NR1fcF Match.Manually matched title.Update.UpdateCopyFiel INF 04-09-09 15:07:53 KODJTH EDBHOC 29357381|B|DR|N
_ 50640355 NR7ulF Report.Cinema report.Fetch.SingleFetch INF 04-09-09 14:29:30 KODJTH EDBFLA 8106517
_ 50640353 NR846F Program.Cue sheet.Fetch.SingleFetchCopyFields INF 04-09-09 14:21:57 KODKMH EDBFLA 36060

F7=Toggle F10=Dump mails F12=Previous F19=Grid cfg
Mere...

M b ↑ 03/011
I902 - Session successfully started
  
```

# Log Data Administration

---

- > Filter on *Program name*, *\*Returning status*, and *Job number*
- > Clear from list when error is corrected
- > View Log data details

# Browsing and Administrating Log Data – Details

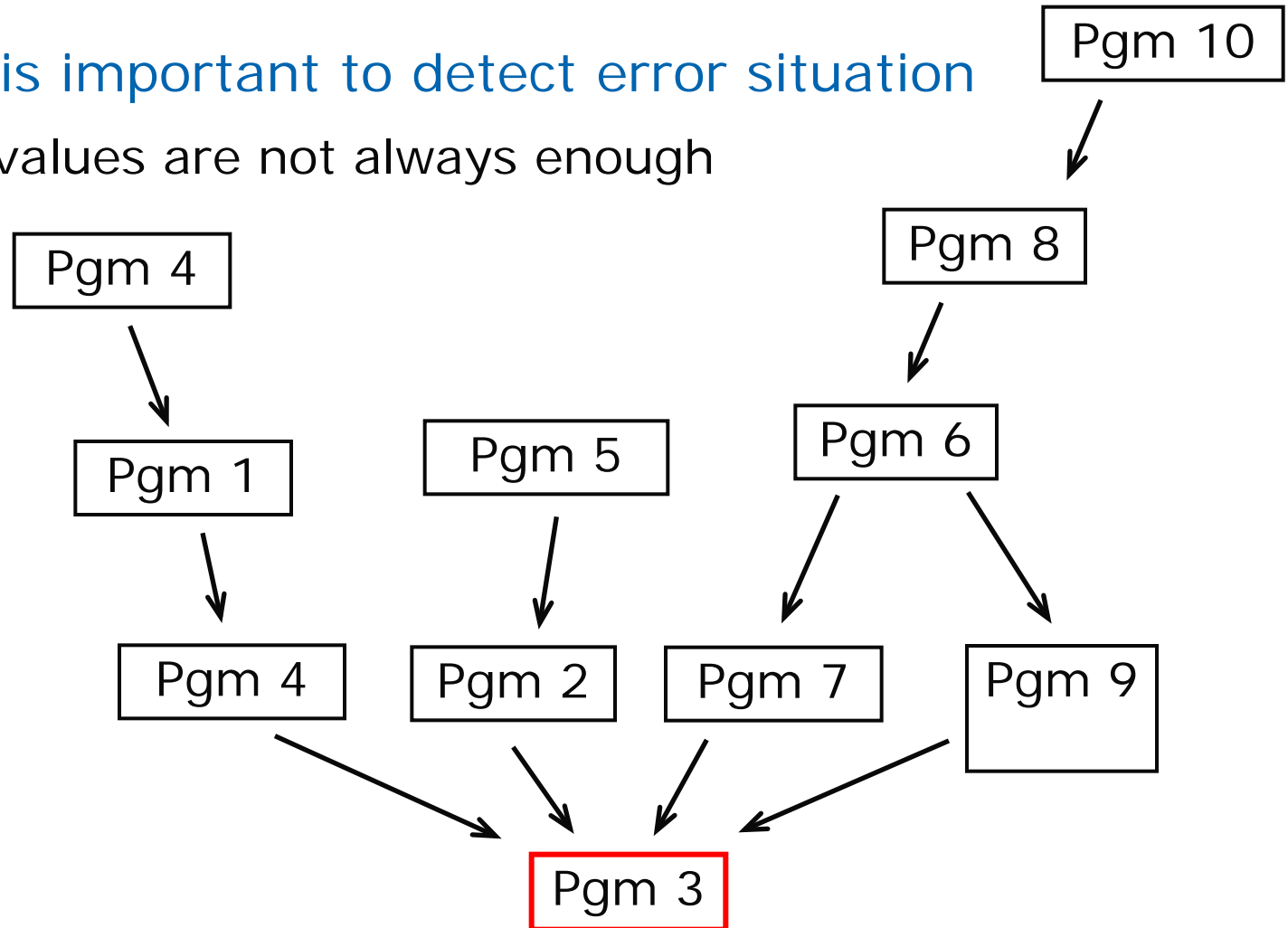
```
Session B - [27 x 132]
File Edit View Communication Actions Window Help
NS12aF Display EDBMKN Display function cal
MShop.Match update
Sequence . . . : 50640373 User . : KODJTH Ret. sts. IEX
Date . : 4-09-09 Time 15:34:24
Development date 1090320160448 Developer EDBFKR
Job number . . : 432612 Job name NRELOAD

Sub# Variable Field Value
-----
1 Specification RunAct M
2 Specification MRecSts M
3 Specification RunSeq 493612000015
4 Source RefSrcId 37232712
5 Target RefTgtId 2056038
```

# Retrieving Program Call Stack

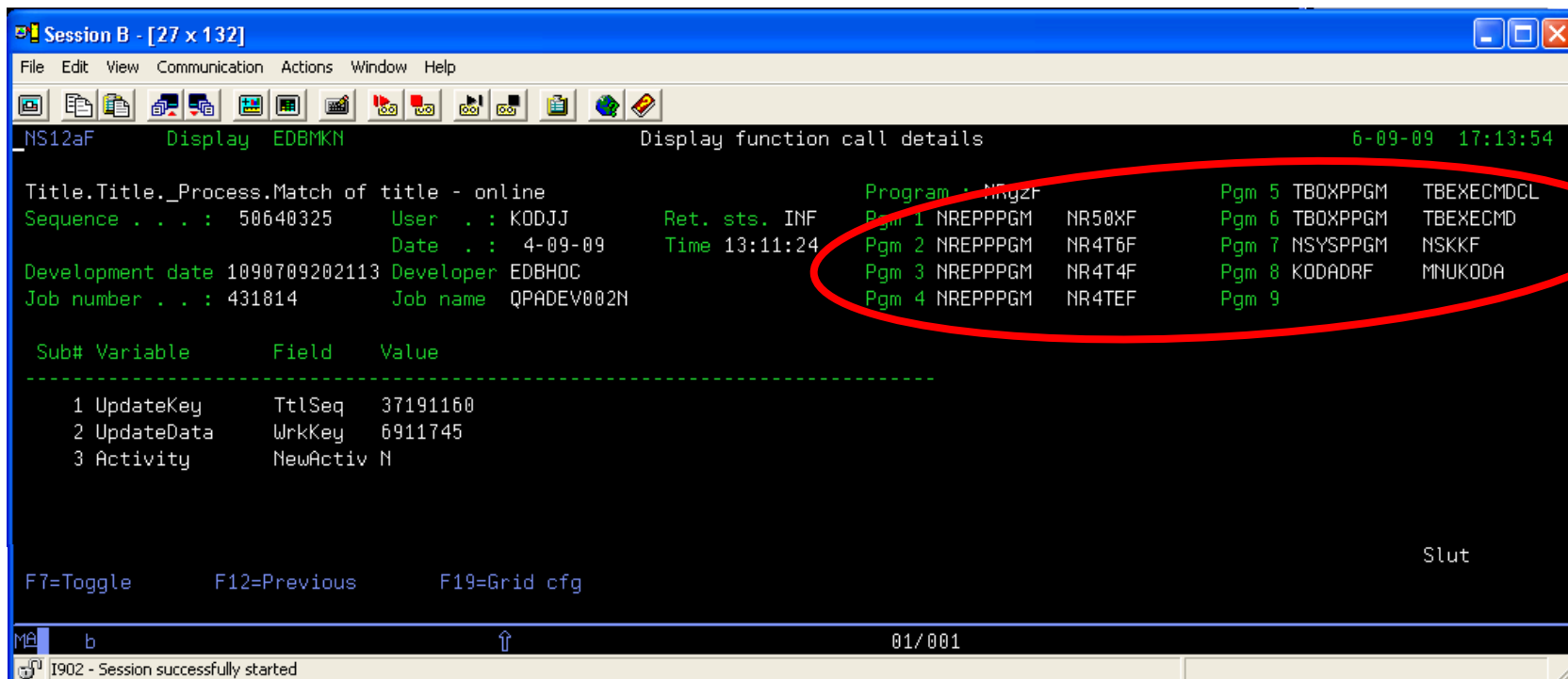
> Call stack is important to detect error situation

- Input values are not always enough



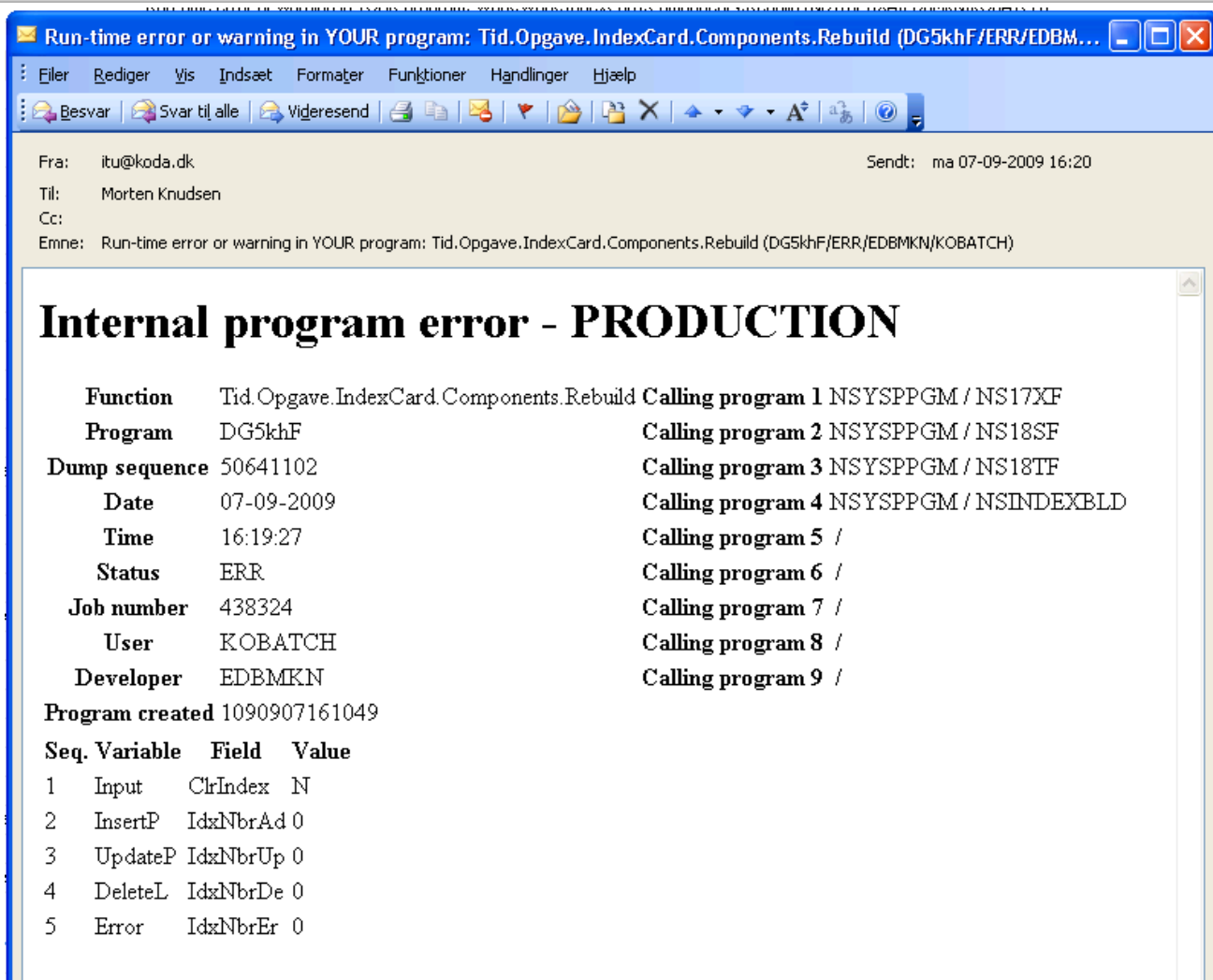
# Retrieving Program Call Stack

- > By retrieving call stack program flow can be followed
  - Call CL program to get names and libraries for latest 9 programs in call stack



```
Session B - [27 x 132]
File Edit View Communication Actions Window Help
NS12aF Display EDBMKN Display function call details 6-09-09 17:13:54
Title.Title._Process.Match of title - online Program : Nrgzf
Sequence . . . : 50640325 User . : KODJJ Ret. sts. INF Pgm 1 NREPPGM NR50XF Pgm 5 TBOXPPGM TBEXECMDCL
Date . : 4-09-09 Time 13:11:24 Pgm 2 NREPPGM NR4T6F Pgm 6 TBOXPPGM TBEXECMD
Developer EDBHOC Pgm 3 NREPPGM NR4T4F Pgm 7 NSYSPPGM NSKKF
Job number . . : 431814 Job name QPADEV002N Pgm 4 NREPPGM NR4TEF Pgm 8 KODADRF MNUKODA
Pgm 9
Sub# Variable Field Value
-----
1 UpdateKey TtlSeq 37191160
2 UpdateData WrkKey 6911745
3 Activity NewActiv N
F7=Toggle F12=Previous F19=Grid cfg Slut
MA b 01/001
I902 - Session successfully started
```

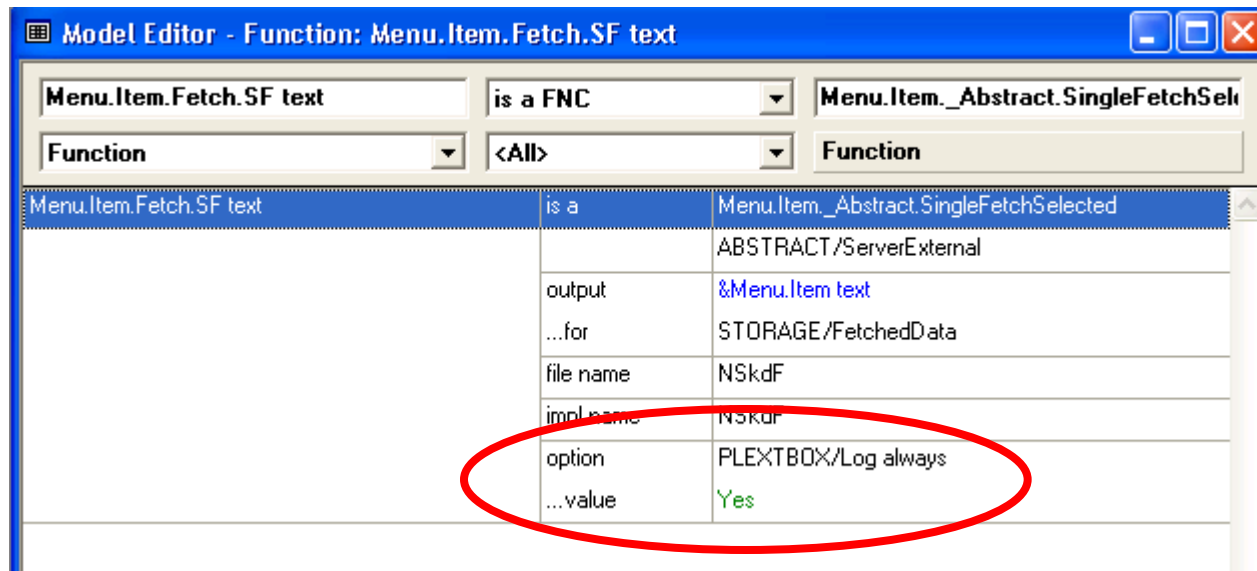
# Email Notification



- > Contains same information as Log details panel
- > Mail send to creator (developer) of program
- > Send to specified administrator
- > Mail only send once per program per date per Returning status

# Log always Option for Debugging

- > Use *Log always* option to dump function input
- > Dump snapshot at various places in program flow
- > Valuable feature for developers not familiar with RPG code and source debugger



- > Feature also implemented for Page Generators and Event handlers
- > Dumping Input variables + *WsyDetails* and *WebInput* respectively

# Extended Logging

Extend the Model to Catch Various Type of Errors

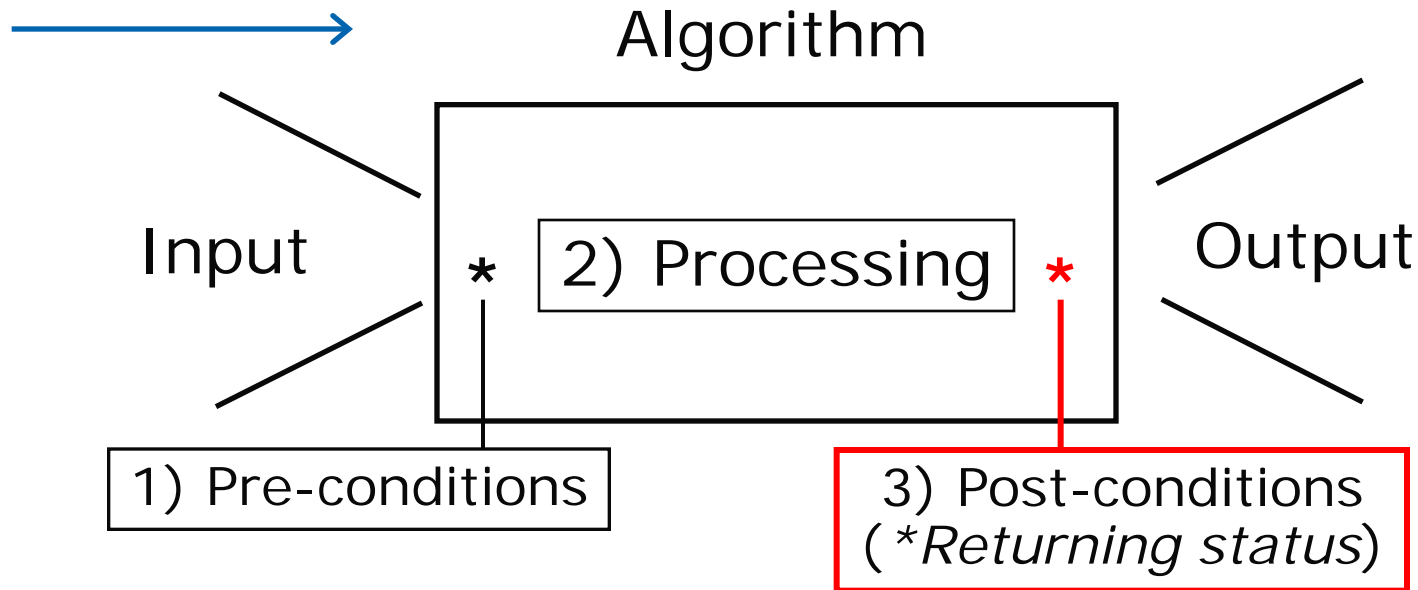
Capture and Log Internal Errors

Log Websyidian Standard Error Messages

Log Communication and Interface Errors

Log Other Application Errors

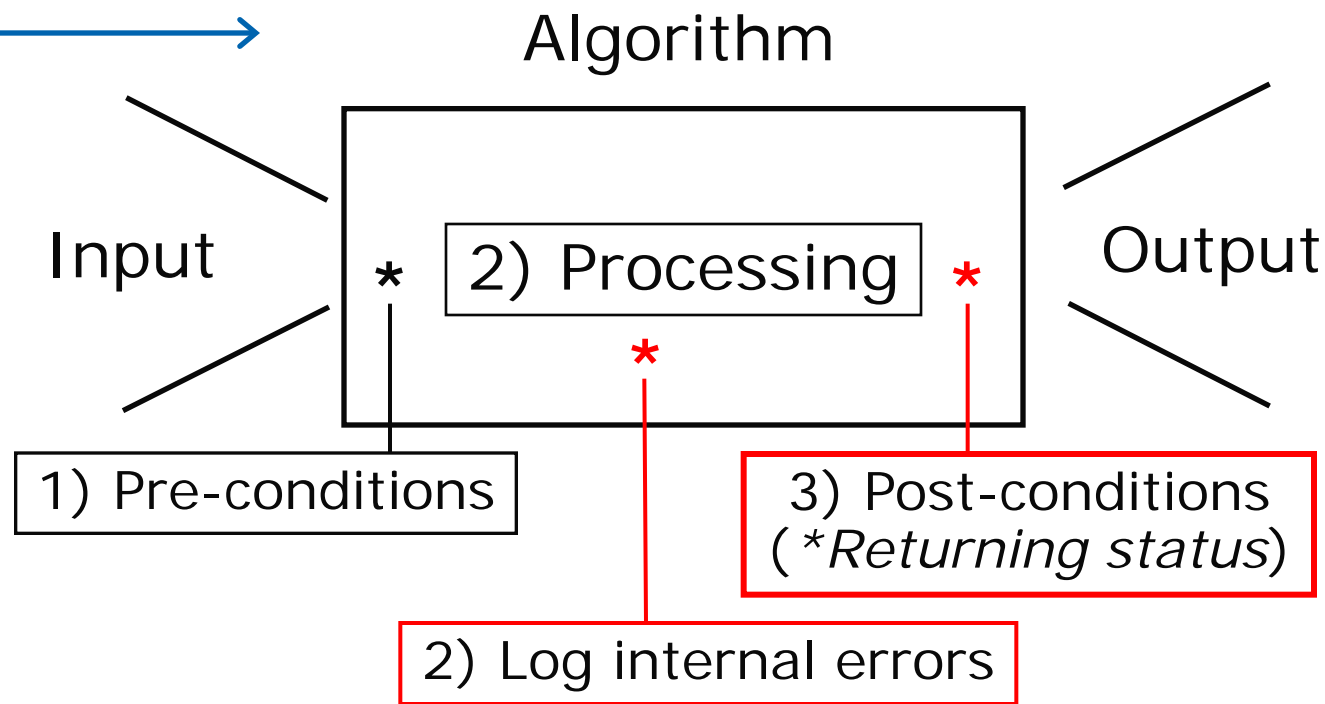
# Using Plex *\*Returning status* to detect abnormal program state



- > Validation of *\*Returning status* post condition
- > Not a complete sanity check

Log when *\*Returning status* is not as expected

# Extending the Model to Catch Various Type of Errors



- > Not only logging of *\*Returning status* post-condition
- > Log various error situations on an abstract level...
- > Special error type assigned to each kind of error

# Capture and Log Internal Errors

- > *YOBPSSR* exception program called by generated program code catching errors 'known by' RPG compiler
  - Invalid data type errors e.g. decimal data error
    - From parameter mapping or *cast* instruction
  - Division by \*zero
  - Overflow
  - Object allocation
  - Further information may be retrieved from joblog...
- > *YOBPSSR* pass all system parameters to Log program
  - Log 'Internal' error

# Capture and Log Internal Errors

Session B - [27 x 132]  
File Edit View Communication Actions Window Help

NS12aF Display EDBMKN Display function call details 7-09-09 23:32:18

Web.User.Fetch.SF user id Program : NSS015F Pgm 5 NSYSPPGM NSYSDISP  
Sequence . . . : 50640688 User . : 16731 Ret. sts. INT Pgm 1 TBOXPPGM YOBPSSR Pgm 6 WSYD560TMP WSYDDRIVCL  
Date . : 5-09-09 Time 23:25:23 Pgm 2 0000003157 NSS015F Pgm 7  
Development date 1090901125840 Developer EDBMKN Pgm 3 NSYSPPGM TCONTPAR Pgm 8  
Job number . . : 435356 Job name NSYSPDISP Pgm 4 NSYSPPGM FCNTFRM Pgm 9

Sub#	Variable	Field	Value
1	Program status	YPPGM	NSS015F
2	Program status	YPPSTS	0
3	Program status	YPPSTP	0
4	Program status	YPPSSQ	00000000
5	Program status	YPPRTN	*INIT
6	Program status	YPPRM	3
7	Program status	YPPMID	
8	Program status	YPPMIN	02CB
9	Program status	YPPMSW	Not able to allocate objects n
10	Program status	YPPPLB	NSYSPPGM
11	Program status	YPPMSD	Not able to allocate objects needed for file USERV1 in library NSYSPDTA member o
12	Program status	YYPEXI	
13	Program status	YYPX01	20
14	Program status	YYPEFL	USERV1

More...

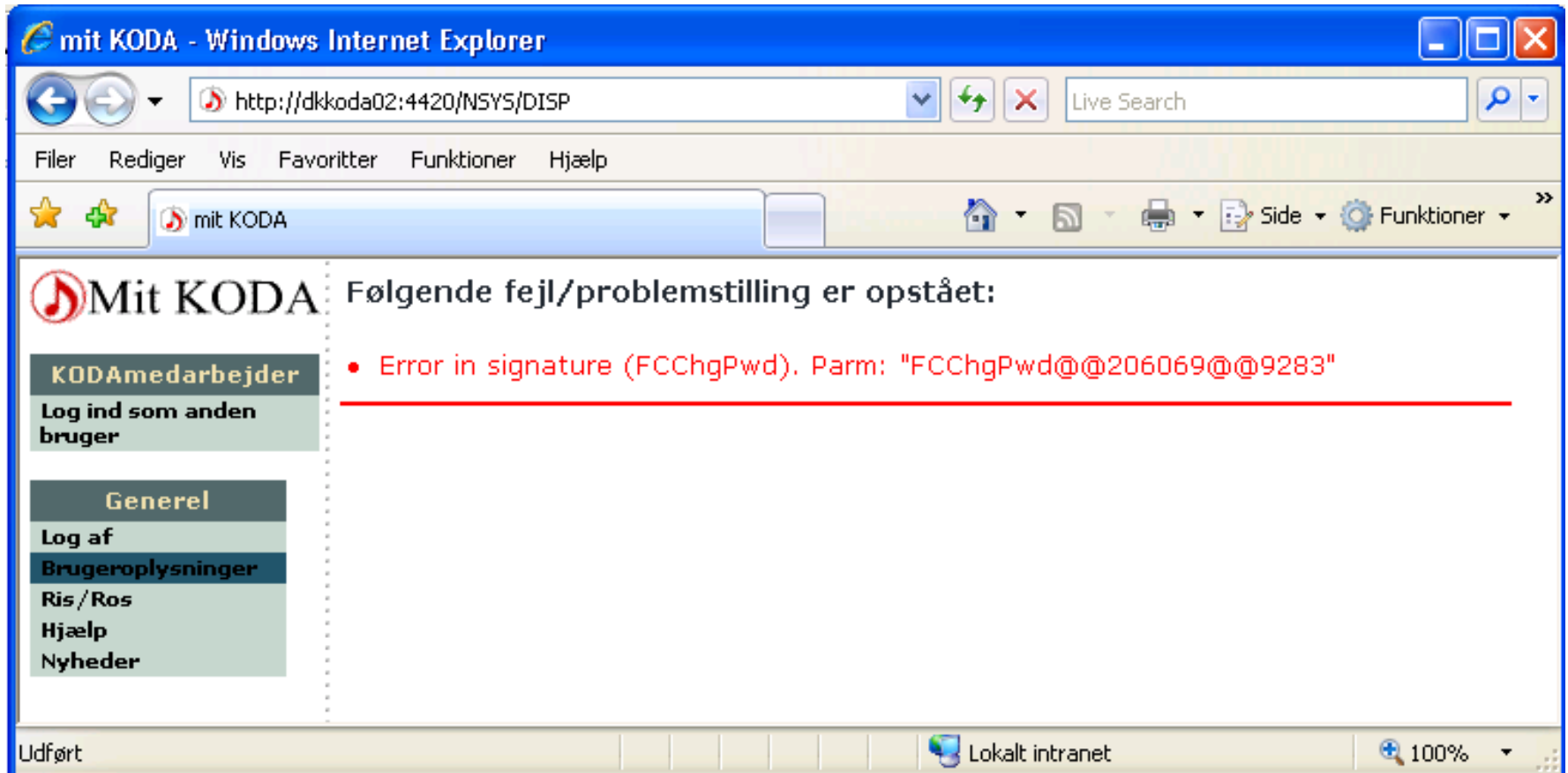
F7=Toggle F12=Previous F19=Grid cfg

MA b ↑ 01/001

I902 - Session successfully started



# Capture and Log Websyidian Standard Errors



# Capture and Log Websyidian Standard Errors

```
Session C - [27 x 132]
File Edit View Communication Actions Window Help
[Icons]
NS12aF Display EDBMKN Display function call details
Web.User.Web.Update User Information.Change Passwo      Program : FCChgPwd      Pgm 5
Sequence . . . : 8883247      User . : 9283      Ret. sts. EVT      Pgm 1 NSYSPPGM      FCCHGPWD      Pgm 6
Date . : 7-09-09      Time 23:44:00      Pgm 2 NSYSPPGM      NSYSDISP      Pgm 7
Development date 1090901125920      Developer EDBMKN      Pgm 3 WSYD560TMP      WSYDDRIVCL      Pgm 8
Job number . . : 433881      Job name NSYSTDISP      Pgm 4
-----
Variable      Field      Value
-----
Input      AbnPgmNm      FCChgPwd
Input      Sqmx1ed      Error in signature (FCChgPwd). Parm: "FCChgPwd@@206069@@9283"
Input      AbnReSts      EVT
```

# Capture and Log Websyidian (Standard) Errors

## > Dispatcher

- Error at create of session
- Session does not exist
- Unable to find event handler
- Unable to call event handler
- No event handler specified
- \*No page generator called by event handler

## > Event handler

- Signature error

# Capture and Log Websyidian (Standard) Errors

## > Page generator

- Unable to open document template
- \*Multiple page generators called by event handler
- Document exceeded cache limit
- Array count exceeded limit

## > Websyidian error logging implemented by calling special function always generating a dump

- Called from abstract specification of *Dispatcher*, *EventHandler* and *PageGenerator* patterns

# Log Communication and Interface Errors

## > KODA Communication/interface Implementations

- Communication with Lucene index server
  - Provides Google-like searching in music work database (containing 20 million records)
  - Errors in index update and retrieval are logged
- Communication with Navision Dynamics database
  - Homemade HTTP protocol
  - Errors in communication are logged

# Log Other Application Errors

---

- > Missing activity reference in history record
- > Negative value not allowed
  
- > ...any kind of error or invalid state that can be detected at an abstract level

# Practical Experience

*\*Returning Status* Logging as Primary  
Source of Error Detection

Lots of Invalid Logs in First Phase of  
Implementation

Errors/Dumps at Batch Processing

# \**Returning Status* Logging as Primary Source of Error Detection

---

- > Used by development
- > Replaced job log as primary error finding source

# Lots of Invalid Logs in First Phase of Implementation

---

- > Error logging was applied to existing applications
  - It took some time to deselect logging not being errors

# Errors/Dumps at Batch Processing...

---

- > Millions of database records may be generated by simple mistakes...

# Questions

---

???