Testing Linked Dyalog APL Workspaces (a work in progress)

Dr. Markos Mitsos
markos.mitsos@ergo.de

Deutsche Krankenversicherung AG DKV - ERGO, Actuarial Department

APL Germany — Stuttgart

ERGO
A Munich Re company

About

Testing Linked Workspaces:

- recap WS-structure und builder
- validate and manage test results
- automated test series

ERGO

A Munich Re company

About

Testing Linked Workspaces:

- recap WS-structure und builder
- validate and manage test results
- automated test series

ERGO

A Munich Re company

About

Testing Linked Workspaces:

- recap WS-structure und builder
- validate and manage test results
- automated test series

ERGO

A Munich Re company

About

Testing Linked Workspaces:

- recap WS-structure und builder
- validate and manage test results
- automated test series

ERGO

A Munich Re company

About

Testing Linked Workspaces:

- recap WS-structure und builder
- validate and manage test results
- automated test series

Necessary tools:

- recap file management (TortoiseSVN)
- test case format (Array Notation)
- test case management (DB2)

ERGO

A Munich Re company

About

Testing Linked Workspaces:

- recap WS-structure und builder
- validate and manage test results
- automated test series

Necessary tools:

- recap file management (TortoiseSVN)
- test case format (Array Notation)
- test case management (DB2)

ERGO

A Munich Re company

About

Testing Linked Workspaces:

- recap WS-structure und builder
- validate and manage test results
- automated test series

Necessary tools:

- recap file management (TortoiseSVN)
- test case format (Array Notation)
- test case management (DB2)

ERGO

A Munich Re company

About

Testing Linked Workspaces:

- recap WS-structure und builder
- validate and manage test results
- automated test series

Necessary tools:

- recap file management (TortoiseSVN)
- test case format (Array Notation)
- test case management (DB2)

ERGO

A Munich Re company

Outline

ERGO

A Munich Re company

Outline

ERGO

A Munich Re company

Outline of section on framework and structure

In this section we outline:

Framework code management and WS structure

Build WS build for coding and debugging

Open open questions and problems

ERGO

A Munich Re company

Outline of section on framework and structure

In this section we outline:

Framework code management and WS structure

Build WS build for coding and debugging

Open open questions and problems

ERGO

A Munich Re company

Outline of section on framework and structure

In this section we outline:

Framework code management and WS structure

Build WS build for coding and debugging

Open open questions and problems

ERGO

A Munich Re company

Outline of section on framework and structure

In this section we outline:

Framework code management and WS structure

Build WS build for coding and debugging

Open open questions and problems

ERGO

A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO

A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO

A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO
A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO
A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO
A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO
A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO

A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO

A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO

A Munich Re company

Target WS framework

What is the framework for coding, debugging,. . . ?

- workspaces
  - Linked (text file based)
  - modular design, “cooperating”
- coding
  - clear and clean distinction code vs. debug
  - decouple code, “save”, test and deploy
- external tools
  - versioning in Tortoise SVN
  - test case management in DB2

ERGO

A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, . . .
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, . . .
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, . . .
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, . . .
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, . . .
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, . . .
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, ...
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, ...
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, ...
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, ...
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, ...
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

ERGO
A Munich Re company

Schematic workspace structure

What is the high level WS structure?

- objects
  - no vars/fns/ops under #
  - main ns <**nsmn**>
  - playground ns **test**
  - building instructions **build**
  - some reserved nss **globals**, **temp**, ...
  - foreign nss <**nsfnX**>
- sources
  - "own" nss primarily from working copy
  - "foreign" nss primarily from repository

▸ WS structure ▸ WS build structure

ERGO
A Munich Re company

WS builds for different purposes

Do I need one WS or many?

- coding needs bi-directionally Linked nss
- debugging should use one-directionally Linked nss
- also use multi-WS debugging
- sometimes use not Linked working copy
- offer use of official checkout as untested deployment

ERGO
A Munich Re company

WS builds for different purposes

Do I need one WS or many?

- coding needs bi-directionally Linked nss
- debugging should use one-directionally Linked nss
- also use multi-WS debugging
- sometimes use not Linked working copy
- offer use of official checkout as untested deployment

ERGO
A Munich Re company

WS builds for different purposes

Do I need one WS or many?

- coding needs bi-directionally Linked nss
- debugging should use one-directionally Linked nss
- also use multi-WS debugging
- sometimes use not Linked working copy
- offer use of official checkout as untested deployment

ERGO
A Munich Re company

WS builds for different purposes

Do I need one WS or many?

- coding needs bi-directionally Linked nss
- debugging should use one-directionally Linked nss
- also use multi-WS debugging
- sometimes use not Linked working copy
- offer use of official checkout as untested deployment

ERGO
A Munich Re company

WS builds for different purposes

Do I need one WS or many?

- coding needs bi-directionally Linked nss
- debugging should use one-directionally Linked nss
- also use multi-WS debugging
- sometimes use not Linked working copy
- offer use of official checkout as untested deployment

ERGO
A Munich Re company

WS builds for different purposes

Do I need one WS or many?

- coding needs bi-directionally Linked nss
- debugging should use one-directionally Linked nss
- also use multi-WS debugging
- sometimes use not Linked working copy
- offer use of official checkout as untested deployment

ERGO
A Munich Re company

WS builds for different purposes

Do I need one WS or many?

- coding needs bi-directionally Linked nss
- debugging should use one-directionally Linked nss
- also use multi-WS debugging
- sometimes use not Linked working copy
- offer use of official checkout as untested deployment

▸ setup for code ▸ setup for debug ▸ standard setup

ERGO
A Munich Re company

Questions and problems

What can be done better?

- replace loading WS with scripting?
- wanted numbered WS logs, maybe named logs through scripting?
- in work
  - occasionally code instability (auto format), especially when code+debug mixed
  - occasionally listeners instability, possibly when coding while running obj for debug
  - cannot grow warm with some aspects of Editor. . .

ERGO

A Munich Re company

Questions and problems

What can be done better?

- replace loading WS with scripting?
- wanted numbered WS logs, maybe named logs through scripting?
- in work
  - occasionally code instability (auto format), especially when code+debug mixed
  - occasionally listeners instability, possibly when coding while running obj for debug
  - cannot grow warm with some aspects of Editor. . .

ERGO
A Munich Re company

Questions and problems

What can be done better?

- replace loading WS with scripting?
- wanted numbered WS logs, maybe named logs through scripting?
- in work
  - occasionally code instability (auto format), especially when code+debug mixed
  - occasionally listeners instability, possibly when coding while running obj for debug
  - cannot grow warm with some aspects of Editor. . .

ERGO
A Munich Re company

Questions and problems

What can be done better?

- replace loading WS with scripting?
- wanted numbered WS logs, maybe named logs through scripting?
- in work
  - occasionally code instability (auto format), especially when code+debug mixed
  - occasionally listeners instability, possibly when coding while running obj for debug
  - cannot grow warm with some aspects of Editor. . .

ERGO
A Munich Re company

Questions and problems

What can be done better?

- replace loading WS with scripting?
- wanted numbered WS logs, maybe named logs through scripting?
- in work
  - occasionally code instability (auto format), especially when code+debug mixed
  - occasionally listeners instability, possibly when coding while running obj for debug
  - cannot grow warm with some aspects of Editor. . .

ERGO
A Munich Re company

Questions and problems

What can be done better?

- replace loading WS with scripting?
- wanted numbered WS logs, maybe named logs through scripting?
- in work
  - occasionally code instability (auto format), especially when code+debug mixed
  - occasionally listeners instability, possibly when coding while running obj for debug
  - cannot grow warm with some aspects of Editor. . .

ERGO
A Munich Re company

Questions and problems

What can be done better?

- replace loading WS with scripting?
- wanted numbered WS logs, maybe named logs through scripting?
- in work
  - occasionally code instability (auto format), especially when code+debug mixed
  - occasionally listeners instability, possibly when coding while running obj for debug
  - cannot grow warm with some aspects of Editor...

ERGO
A Munich Re company

Outline of section on test and deployment

In this section we outline:

- Framework: test construction, validation and repetition
- Examples: target, check and deployment
- Open: open questions and problems

ERGO

A Munich Re company

Outline of section on test and deployment

In this section we outline:

- Framework test construction, validation and repetition
- Examples target, check and deployment
- Open open questions and problems

ERGO

A Munich Re company

Outline of section on test and deployment

In this section we outline:

- Framework test construction, validation and repetition
- Examples target, check and deployment
- Open open questions and problems

ERGO

A Munich Re company

Outline of section on test and deployment

In this section we outline:

- **Framework** test construction, validation and repetition
- **Examples** target, check and deployment
- **Open** open questions and problems

ERGO

A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE WS`
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO
A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE WS`
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO
A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE WS`
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO
A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE WS`
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO
A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE WS`
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO
A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE` WS
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO
A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE` WS
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO
A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE` WS
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO
A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE` WS
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO

A Munich Re company

Target test framework

What is the framework for debugging, testing,. . . ?

- debug and remember it
  - build test cases, try them out
  - when OK save arguments and results as target (SOLL)
- deploy if sure everything is OK
  - run all test cases, then `)SAVE` WS
  - document comparison between actual (IST) and target for IDP audit
- check when Dyalog or environment changes
  - run at least tests on base utilities when environment changes
  - document comparison for release audit

ERGO
A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO

A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO

A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO

A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO

A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO
A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO

A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO
A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO

A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO
A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO
A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO
A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO
A Munich Re company

"Unit" tests and period system time tables

What kind of tests are used?

- "unit" test cases
  - want defined in- and output
  - but output too large to explicitly write down
- structure
  - use **build.check** to collect test case
  - one fn for each ns
  - one `:CASE` for each obj
  - remove/replace non-deterministic results prior to saving
- save in DB2
  - period system time very useful concept
  - implicitly hidden also useful
- (partly) show in Excel (for "externals")

ERGO
A Munich Re company

Staged comparison

All or nothing approach not practical

- test case description as key
- non-identifiable cases?
- try also identification through object call?
- different object calls?
- different results?
- different sorting?

ERGO
A Munich Re company

Staged comparison

All or nothing approach not practical

- test case description as key
- non-identifiable cases?
- try also identification through object call?
- different object calls?
- different results?
- different sorting?

ERGO
A Munich Re company

Staged comparison

All or nothing approach not practical

- test case description as key
- non-identifiable cases?
- try also identification through object call?
- different object calls?
- different results?
- different sorting?

ERGO
A Munich Re company

Staged comparison

All or nothing approach not practical

- test case description as key
- non-identifiable cases?
- try also identification through object call?
- different object calls?
- different results?
- different sorting?

ERGO
A Munich Re company

Staged comparison

All or nothing approach not practical

- test case description as key
- non-identifiable cases?
- try also identification through object call?
- different object calls?
- different results?
- different sorting?

ERGO

A Munich Re company

Staged comparison

All or nothing approach not practical

- test case description as key
- non-identifiable cases?
- try also identification through object call?
- different object calls?
- different results?
- different sorting?

ERGO

A Munich Re company

Staged comparison

All or nothing approach not practical

- test case description as key
- non-identifiable cases?
- try also identification through object call?
- different object calls?
- different results?
- different sorting?

ERGO
A Munich Re company

Use comparison in function

Implement in #.div.buld.CHECK

- validation
  - separate modi
  - throw error if deviation above threshold
  - special modus for description change
- check
  - comparison creates flagged results

ERGO
A Munich Re company

Use comparison in function

Implement in #.div.buld.CHECK

- validation
  - separate modi
  - throw error if deviation above threshold
  - special modus for description change
- check
  - comparison creates flagged results

ERGO
A Munich Re company

Use comparison in function

Implement in #.div.buld.CHECK

- validation
  - separate modi
  - throw error if deviation above threshold
  - special modus for description change
- check
  - comparison creates flagged results

ERGO
A Munich Re company

Use comparison in function

Implement in #.div.buld.CHECK

- validation
  - separate modi
  - throw error if deviation above threshold
  - special modus for description change
- check
  - comparison creates flagged results

ERGO
A Munich Re company

Use comparison in function

Implement in #.div.buld.CHECK

- validation
  - separate modi
  - throw error if deviation above threshold
  - special modus for description change
- check
  - comparison creates flagged results

ERGO
A Munich Re company

Use comparison in function

Implement in #.div.buld.CHECK

- validation
  - separate modi
  - throw error if deviation above threshold
  - special modus for description change
- check
  - comparison creates flagged results

ERGO
A Munich Re company

Use comparison in function

Implement in #.div.buld.CHECK

- validation
  - separate modi
  - throw error if deviation above threshold
  - special modus for description change
- check
  - comparison creates flagged results

ERGO
A Munich Re company

WS LATEX

Test interface between Dyalog and LaTeX

- ns **tex** is main ns of LATEX
- ns **tex.imp** contains objs for importing TeX files
- ARG_ANALYSIEREN parses arguments of an LaTeX macro
- CMD_ANALYSIEREN parses document concerning a list of LaTeX macros

ERGO
A Munich Re company

WS LATEX

Test interface between Dyalog and LaTeX

- ns **tex** is main ns of LATEX
- ns **tex.imp** contains objs for importing TeX files
- ARG_ANALYSIEREN parses arguments of an LaTeX macro
- CMD_ANALYSIEREN parses document concerning a list of LaTeX macros

ERGO

A Munich Re company

WS LATEX

Test interface between Dyalog and LaTeX

- ns **tex** is main ns of LATEX
- ns **tex.imp** contains objs for importing TeX files
- ARG_ANALYSIEREN parses arguments of an LaTeX macro
- CMD_ANALYSIEREN parses document concerning a list of LaTeX macros

ERGO
A Munich Re company

WS LATEX

Test interface between Dyalog and LaTeX

- ns **tex** is main ns of LATEX
- ns **tex.imp** contains objs for importing TeX files
- ARG_ANALYSIEREN parses arguments of an LaTeX macro
- CMD_ANALYSIEREN parses document concerning a list of LaTeX macros

ERGO
A Munich Re company

WS LATEX

Test interface between Dyalog and LaTeX

- ns **tex** is main ns of LATEX
- ns **tex.imp** contains objs for importing TeX files
- ARG_ANALYSIEREN parses arguments of an LaTeX macro
- CMD_ANALYSIEREN parses document concerning a list of LaTeX macros

ERGO
A Munich Re company

Argument parsing ARG_ANALYSIEREN

Test parsing of arguments

- in- and output of fn small, could be written down
- contains range of checks on arguments (also to be tested)
- fn meant as subfunction
- needed because of recursion (`\section{Abschnitt}` vs. `\rbw[\alter[0]][np]`)

ERGO
A Munich Re company

Argument parsing ARG_ANALYSIEREN

Test parsing of arguments

- in- and output of fn small, could be written down
- contains range of checks on arguments (also to be tested)
- fn meant as subfunction
- needed because of recursion (`\section{Abschnitt}` vs. `\rbw[\alter[0]][np]`)

ERGO
A Munich Re company

Argument parsing ARG_ANALYSIEREN

Test parsing of arguments

- in- and output of fn small, could be written down
- contains range of checks on arguments (also to be tested)
- fn meant as subfunction
- needed because of recursion (\section{*Abschnitt*} vs. \rbw[*alter[0]*][*np*])

ERGO
A Munich Re company

Argument parsing ARG_ANALYSIEREN

Test parsing of arguments

- in- and output of fn small, could be written down
- contains range of checks on arguments (also to be tested)
- fn meant as subfunction
- needed because of recursion (`\section{Abschnitt}` vs. `\rbw[\alter[0]][np]`)

ERGO
A Munich Re company

Argument parsing ARG_ANALYSIEREN

Test parsing of arguments

- in- and output of fn small, could be written down
- contains range of checks on arguments (also to be tested)
- fn meant as subfunction
- needed because of recursion (\section{*Abschnitt*} vs. \rbw[*alter[0]*][*np*])

ERGO
A Munich Re company

Document parsing CMD_ANALYSIEREN

Test parsing of documents

- needs list of macros, write down
- needs document in special form, better keep in TeXstudio and import on the fly
- result can be lengthy and complicated

ERGO
A Munich Re company

Document parsing CMD_ANALYSIEREN

Test parsing of documents

- needs list of macros, write down
- needs document in special form, better keep in TeXstudio and import on the fly
- result can be lengthy and complicated

ERGO
A Munich Re company

Document parsing CMD_ANALYSIEREN

Test parsing of documents

- needs list of macros, write down
- needs document in special form, better keep in TeXstudio and import on the fly
- result can be lengthy and complicated

ERGO
A Munich Re company

Document parsing CMD_ANALYSIEREN

Test parsing of documents

- needs list of macros, write down
- needs document in special form, better keep in TeXstudio and import on the fly
- result can be lengthy and complicated

ERGO
A Munich Re company

Self test of #.div.buld.CHECK

Check check function itself

- create temporal namespaces
- populate with test function, test cases and corresponding list
- use separate DB2 creator

ERGO
A Munich Re company

Self test of #.div.buld.CHECK

Check check function itself

- create temporal namespaces
- populate with test function, test cases and corresponding list
- use separate DB2 creator

ERGO
A Munich Re company

Self test of #.div.buld.CHECK

Check check function itself

- create temporal namespaces
- populate with test function, test cases and corresponding list
- use separate DB2 creator

ERGO

A Munich Re company

Self test of #.div.buld.CHECK

Check check function itself

- create temporal namespaces
- populate with test function, test cases and corresponding list
- use separate DB2 creator

ERGO

A Munich Re company

Questions and problems

What can be done better?

- went to *CLOB(4M)*, but cannot select
- use pseudo UTF-16 encoding to circumvent
- format via Array Notation, but some problems with cycle
- result too long, even up to WS FULL
- also deploy other WS forms (runtime, dll,...)?

ERGO
A Munich Re company

Questions and problems

What can be done better?

- went to *CLOB(4M)*, but cannot select
- use pseudo UTF-16 encoding to circumvent
- format via Array Notation, but some problems with cycle
- result too long, even up to WS FULL
- also deploy other WS forms (runtime, dll,...)?

ERGO
A Munich Re company

Questions and problems

What can be done better?

- went to *CLOB(4M)*, but cannot select
- use pseudo UTF-16 encoding to circumvent
- format via Array Notation, but some problems with cycle
- result too long, even up to WS FULL
- also deploy other WS forms (runtime, dll,...)?

ERGO
A Munich Re company

Questions and problems

What can be done better?

- went to *CLOB(4M)*, but cannot select
- use pseudo UTF-16 encoding to circumvent
- format via Array Notation, but some problems with cycle
- result too long, even up to WS FULL
- also deploy other WS forms (runtime, dll,...)?

ERGO
A Munich Re company

Questions and problems

What can be done better?

- went to *CLOB(4M)*, but cannot select
- use pseudo UTF-16 encoding to circumvent
- format via Array Notation, but some problems with cycle
- result too long, even up to WS FULL
- also deploy other WS forms (runtime, dll,...)?

ERGO
A Munich Re company

Questions and problems

What can be done better?

- went to *CLOB(4M)*, but cannot select
- use pseudo UTF-16 encoding to circumvent
- format via Array Notation, but some problems with cycle
- result too long, even up to WS FULL
- also deploy other WS forms (runtime, dll,...)?

ERGO
A Munich Re company

Conclusion

Open problems:

- scripting and multiple logs
- code stability and Editor behaviour (getting better)
- bigger arguments/results in DB2 (almost solved)
- format of test cases and Array Notation (some points remaining)
- deployment of more WS variants

ERGO

A Munich Re company

Conclusion

Open problems:

- scripting and multiple logs
- code stability and Editor behaviour (getting better)
- bigger arguments/results in DB2 (almost solved)
- format of test cases and Array Notation (some points remaining)
- deployment of more WS variants

ERGO

A Munich Re company

Conclusion

Open problems:

- scripting and multiple logs
- code stability and Editor behaviour (getting better)
- bigger arguments/results in DB2 (almost solved)
- format of test cases and Array Notation (some points remaining)
- deployment of more WS variants

ERGO
A Munich Re company

Conclusion

Open problems:

- scripting and multiple logs
- code stability and Editor behaviour (getting better)
- bigger arguments/results in DB2 (almost solved)
- format of test cases and Array Notation (some points remaining)
- deployment of more WS variants

ERGO

A Munich Re company

Conclusion

Open problems:

- scripting and multiple logs
- code stability and Editor behaviour (getting better)
- bigger arguments/results in DB2 (almost solved)
- format of test cases and Array Notation (some points remaining)
- deployment of more WS variants

ERGO

A Munich Re company

Conclusion

Open problems:

- scripting and multiple logs
- code stability and Editor behaviour (getting better)
- bigger arguments/results in DB2 (almost solved)
- format of test cases and Array Notation (some points remaining)
- deployment of more WS variants

ERGO
A Munich Re company

Conclusion

Open problems:

- scripting and multiple logs
- code stability and Editor behaviour (getting better)
- bigger arguments/results in DB2 (almost solved)
- format of test cases and Array Notation (some points remaining)
- deployment of more WS variants

begin

ERGO
A Munich Re company

Overview of examples and illustrations

- WS structure
- WS build structure
- setup for code
- setup for debug
- standrad setup

ERGO

A Munich Re company

Schematic structure of workspace

#.<nsmn>	code proper of workspace
#.build	building instructions and tests cases
#.test	alternatives, ideas,. . .
#.<nsfnX>	imported foreign namespace[s]

#.temp	temporary, for example during build
#.globals	global object, for example COM

◂ Schematic workspace structure

ERGO
A Munich Re company

Schematic structure of namespace #.build.prms

nsl	source and target of relevant namespaces
wsn	target name of deployed workspace
lx	latent expression of deployed workspace
nsmn	main "own" namespace of workspace

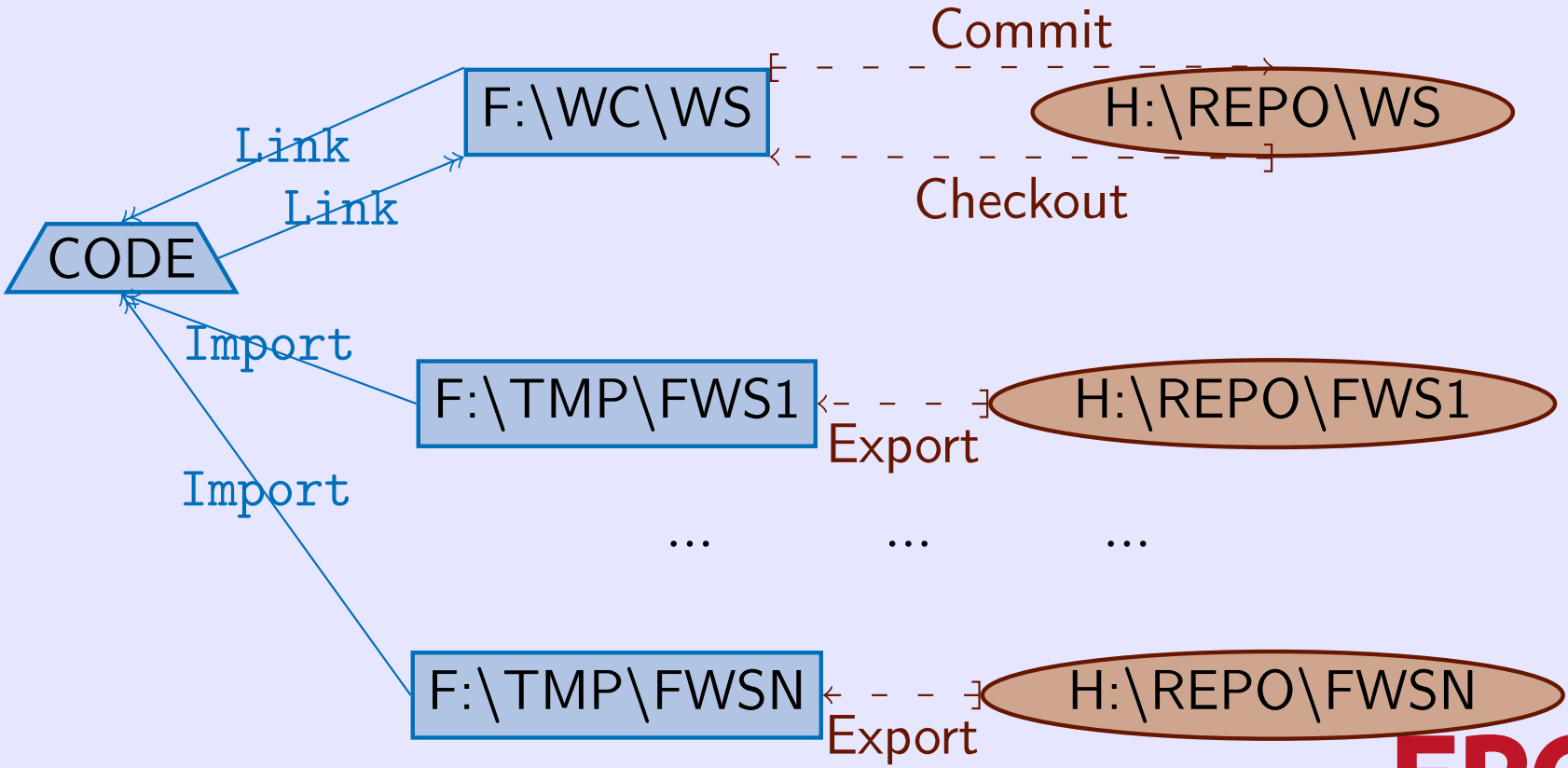
prvdr	DB2 provider for test case management
sqlid	DB2 SQLID for test case management
xln	Excel target for test validation documentation

◂ Schematic workspace structure

ERGO
A Munich Re company

Code workspace

Own working copy is Linked bi-directionaly, Rest Imported:



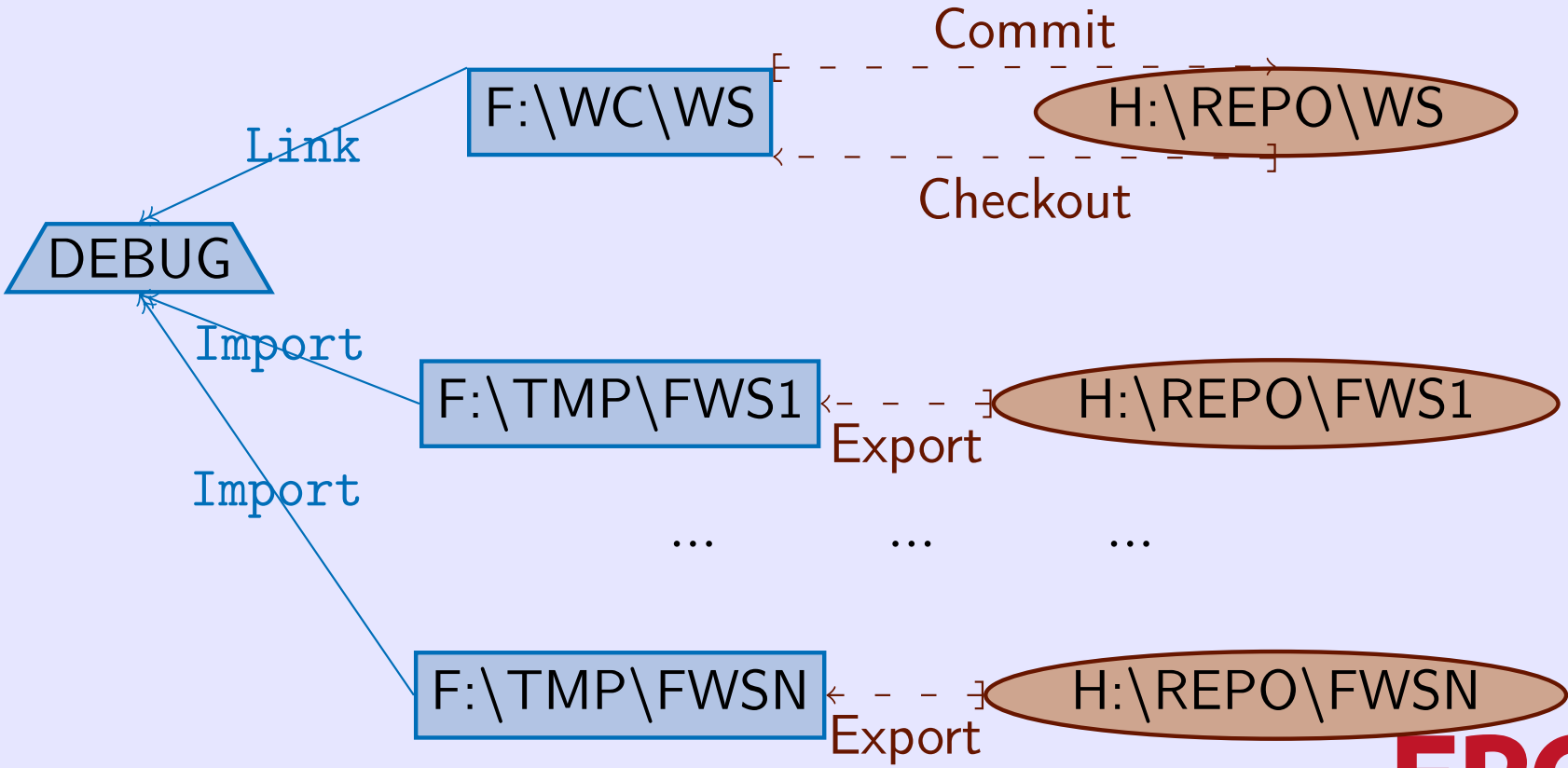
◂ WS build for coding

ERGO

A Munich Re company

Debug workspace

Own working copy is Linked uni-directionaly, Rest Imported:



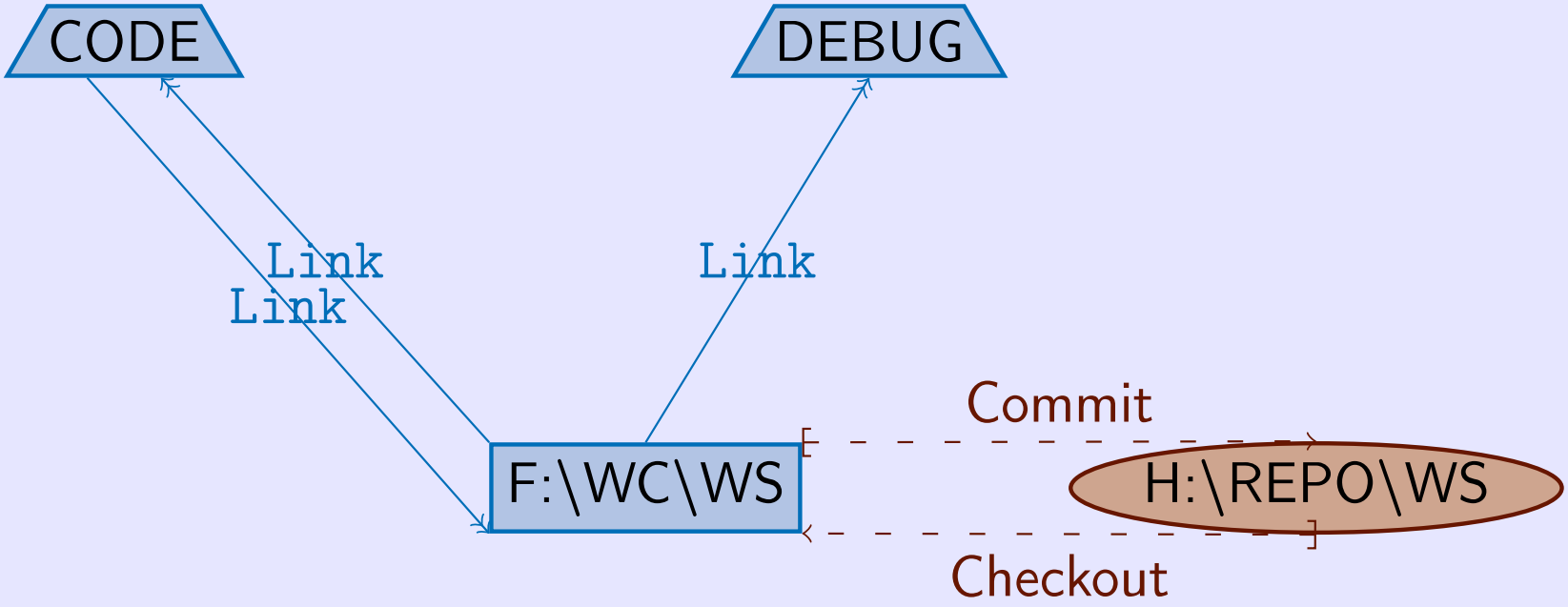
◂ WS build for debugging

ERGO

A Munich Re company

Code and debug workspace

Usual work setup, two WS in tandem:



◂ WS build for debugging

ERGO

A Munich Re company