2nd Annual
MidPoint Community Meetup

Let's together
SAY GOODBYE TO "SCRAPPYCODE"

Evolveum

Funded by the European Union
NextGenerationEU

RECOVERY AND RESILIENCE PLAN

Pavol Mederly, May 2026
Interim Chief Product Officer

Current Situation

- Configuration = XML + **scripts**
  - short or long / simple or complex
- Development and **maintenance** burden
  - take time to create and debug
  - then to (re)**understand** and **adapt** (business/technical reasons)
- Our goal: **reduce the effort!**

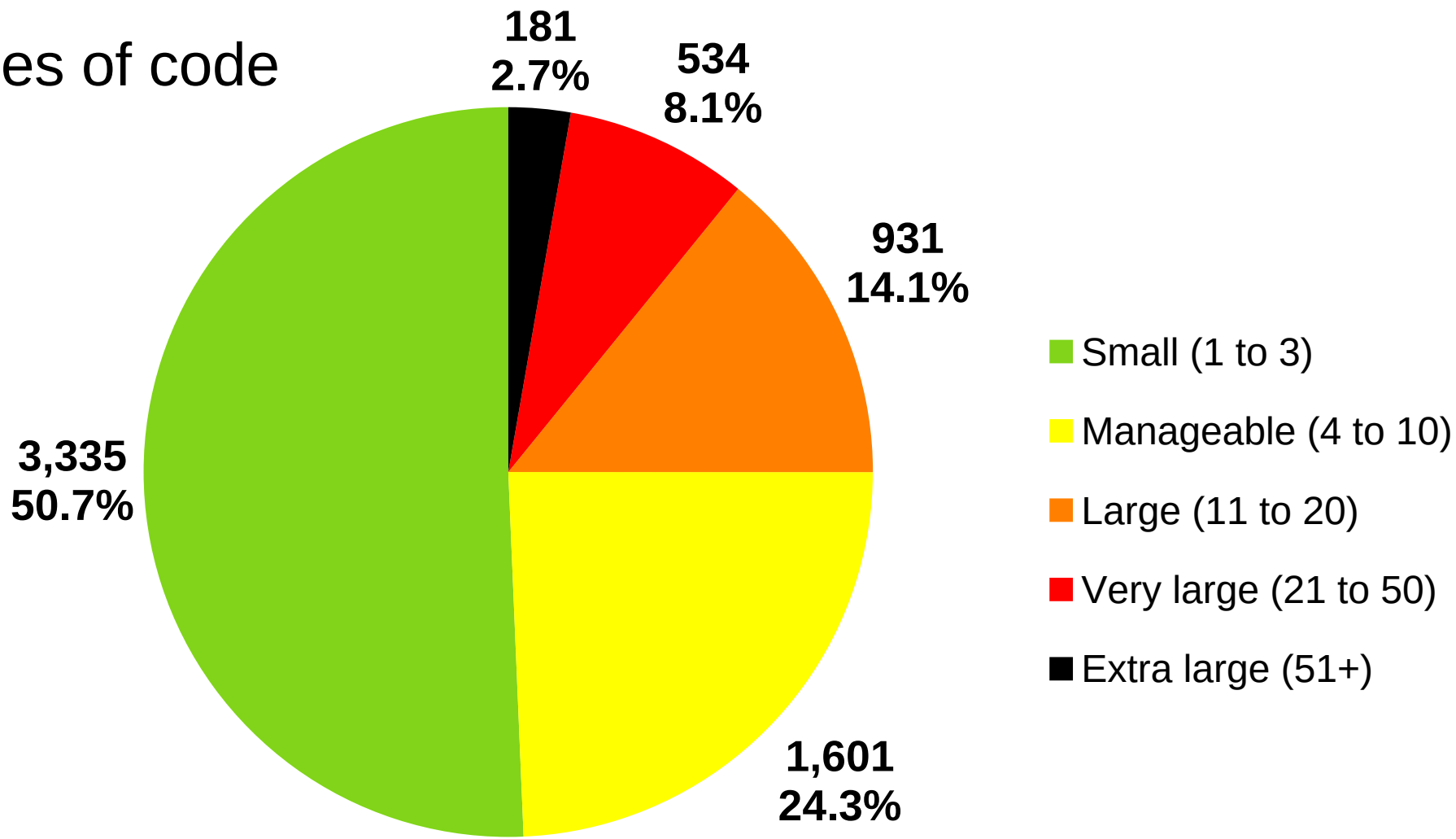
Analysis Done

- Informal knowledge
  - decades of experience
  - supported by 50 years of software engineering knowledge
  - and discussions within the community
- Hard data from **10 deployments**
  - **6821 scripts** from 1174 files
  - analyzed manually and by LLM
  - using 50 tags

Looking for...

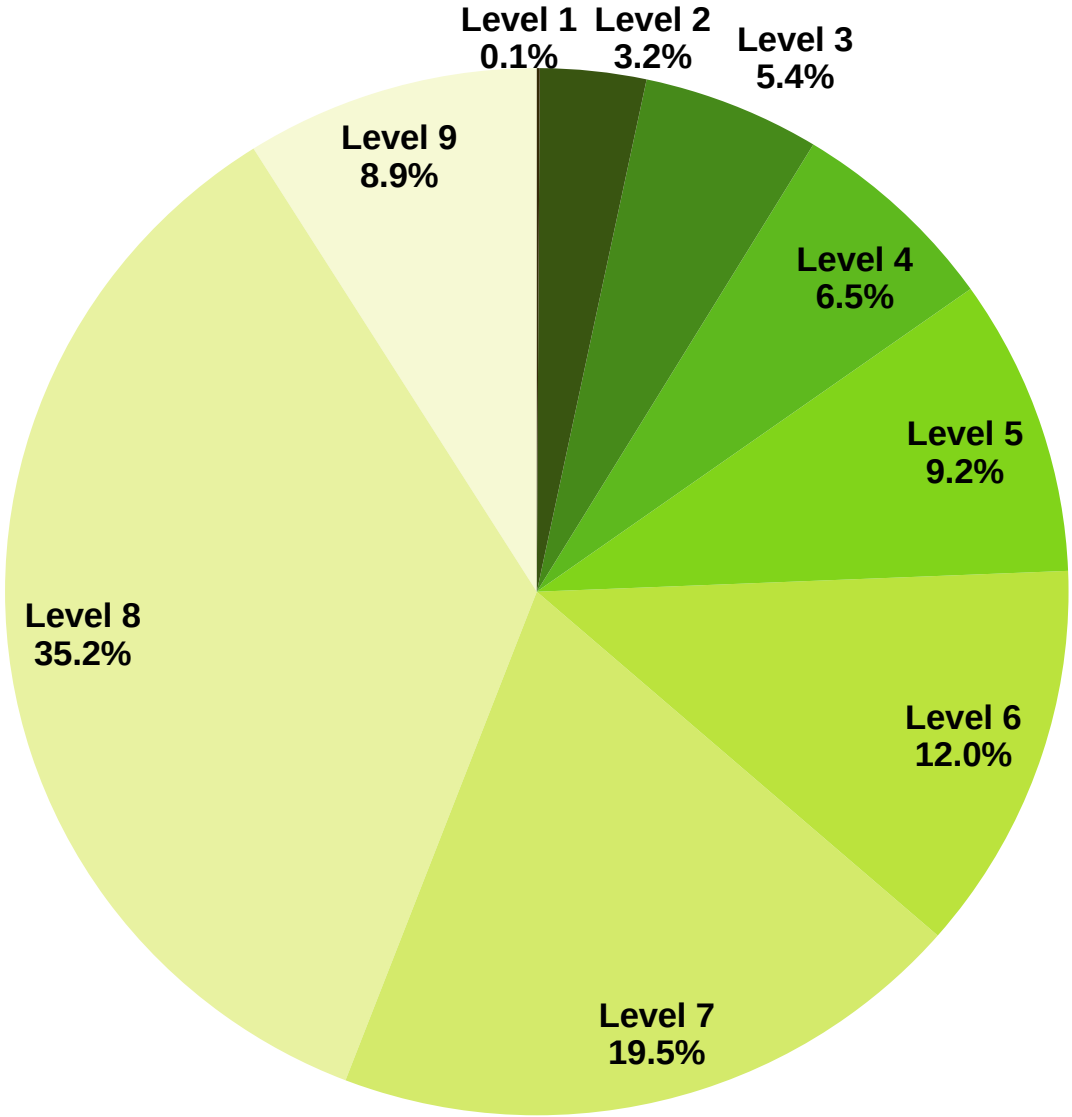
- New midPoint features → to **remove** the scripts
  - new expression evaluators (like `assignmentTargetSearch`)
  - new features (like “complex attributes”)
- New API methods → to **simplify** the scripts

Lines of code



Maintainability (estimated)

- L9: simple string concatenation; a static value; null-check expression; simple checks combined by logical AND; …
- L8: simple conditional with a clear intent; single, clear method call; …
- L7: straightforward code but with unnecessary steps; short script with a logic that is a bit opaque; …
- …
- **L3:** dense, nested boolean expressions with magic strings and undocumented variables; …
- **L2:** many magic strings, nested and compound conditionals, and unclear variable names; …
- **L1:** extensive duplicated logic for multiple regexps, deep nesting, inconsistent naming, and no abstraction; ...



Suggestions: Areas of Improvement

- Making business logic explicit
- Constants for OIDs and other strings
- Assignment processing
- Archetype matching
- Date/time manipulation
- Text normalization
- Notification text formatting
- Delta analysis
- Creating objects
- Manipulating names, emails, phone numbers, LDAP DNs
- Complex attributes
- Easier getting extension items and attributes + calling function libraries
- Handling owners, managers, approvers
- Decision trees and mapping tables
- Pretty-printing and diagnostics
- Organizational tree walking

Making Business Logic Explicit

- Instead of writing **technical code** dealing with archetypes, extension properties, ...
  - `midpoint.hasArchetype(focus, 'f309eb68-2a78-44bd-aaa8-085fbc49ef9c')`
  - `basic.getExtensionPropertyValue(focus, new QName("http://example.com/", "studentId")`
- ...we can define **custom business-oriented functions** for given object type (or even archetype) or assignment like:
  - `focus.isEmployee()`
  - `focus.getStudentId()`
  - `assignment.getDepartmentName()`
  - `assignment.getStudyProgramId()`
- This feature alone will lead to **significant simplification of scripting code**
- Applicable to MEL and Groovy

Assignment Processing (100+ cases)

```
focus.assignment?.findAll { it.targetRef?.type?.localPart == "ServiceType" }?.each { assignment ->
    def service = midpoint.getObject(ServiceType.class, assignment.targetRef.oid)
    if (midpoint.hasArchetype(service, "********-****-****-****-************")) {
        def activation = service.activation
        def status = activation?.effectiveStatus
        if (status == ActivationStatusType.ENABLED) {
            activeServices << service.oid
        } else {
            inactiveServices << service.oid
        }
    }
}
```

- `targetsRole(), targetsOrg(), targetsService(), targetsArchetype(), ...`
- `targetsObjectWithArchetype(Archetype.STUDY_PROGRAM)`
- `hasDefaultRelation(), hasManagerRelation(), hasOwnerRelation(), ...`
- `targets(Role.ACCOUNTANT)`
- `targets { it.name == 'Accountant' }`
- `targets { it.costCenter == '123456' }`
- `target()`

- ```
  focus.directAssignments
    .findAll { it.targetsRoleWithArchetype(Archetype.SYSTEM_ROLE) }
  ```
- ```
  focus.indirectAssignments
    .findAll { it.targetsRole() && it.hasManagerRelation() }
  ```
- `focus.directAssignments.any { it.targets(Role.ACCOUNTANT) }`
- `focus.directAssignments.any { it.targets(Role.ACCOUNTANT) && it.hasManagerRelation() }`

Assignment Processing: A Real Example

```
focus.assignment?.findAll { it.targetRef?.type?.localPart == "ServiceType" }?.each { assignment ->
    def service = midpoint.getObject(ServiceType.class, assignment.targetRef.oid)
    if (midpoint.hasArchetype(service, "********-****-****-****-************")) {
        def activation = service.activation
        def status = activation?.effectiveStatus
        if (status == ActivationStatusType.ENABLED) {
            activeServices << service.oid
        } else {
            inactiveServices << service.oid
        }
    }
}
```

```
focus.directAssignments
    .findAll { it.targetsServiceWithArchetype(Archetype.APPLICATION) }
    .each {
        if (it.target().isEnabled) {
            activeServices << it.oid
        } else {
            inactiveServices << it.oid
        }
    }
```

Archetype Matching

- Focus (or other object) has given archetype or archetypes
  - by OID, by name, or by other criteria
- Considering
  - `focus.hasArchetype(Archetype.EMPLOYEE)`
  - `focus.hasAnyArchetype(Archetype.EMPLOYEE, Archetype.STUDENT)`
  - `focus.hasAllArchetypes(Archetype.EMPLOYEE, Archetype.INTERNAL)`
- ...but generally this is better:
  - `focus.isEmployee() || focus.isStudent()`
  - `focus.isInternalEmployee()`

Date/Time Manipulation (450+ cases)

- date/time conversion: strings + other formats; including Unix/AD timestamps
- date/time comparison
- duration arithmetic
- rounding: end of day, week, month, …

```
if (statusOverride) {
    return 122891712000000000
} else {
    time = MiscUtil.asDate(validTo)?.getTime()
    return time * 10000 + 116444736000000000
}
```

```
statusOverride ? basic.farAhead() : validTo
```

```
if (validFrom) {
    DateFormat fmt = new SimpleDateFormat( pattern: 'YYYY-MM-dd HH:mm:ss')
    return fmt.format(validFrom.toGregorianCalendar().getTime()) + '.0'
} else {
    return null;
}
```

```
basic.formatTime(validFrom, 'YYYY-MM-dd HH:mm:ss.0')
```

Text Normalization

- MidPoint has built-in mechanisms for text normalization
- … but custom algorithms seem to be present in every deployment!
- We consider improving the built-in configurable normalizer based on what we saw in the projects

```
def map = [ 'À' : 'A',
            'Á' : 'A',
            'Â' : 'A',
            'Ã' : 'A',
            'Ä' : 'Ae',
            'Å' : 'Aa',
            'Æ' : 'AE',
            'Ç' : 'C',
            'È' : 'E',
            'É' : 'E',
            'Ê' : 'E',
            'Ë' : 'E',
```

```
def charMap = [
        "ß": "ss",
        "Ä": "Ae",
        "Ö": "Oe",
        "Ü": "Ue",
        "ä": "ae",
        "ö": "oe",
        "ü": "ue",
        "Å": "A",
        "å": "a",
        "Ç": "C",
        "ç": "c",
        "Ñ": "N",
        "ñ": "n"]
```

Notification Text Formatting (ASCII/HTML) (200+ cases)

- MidPoint supports Apache Velocity as an expression language
- … yet a lot of custom code is usually present to generate notifications
- It seems that we need a simple way of integrating templates with the custom Groovy/MEL code
- Including configurable/extensible delta and object data formatters

```
import java.text.SimpleDateFormat
import java.text.DateFormat
String fileName
fileName = "/opt/midpoint/var/source/my-task-notification-template.html"
File file = new File(fileName)
String template = file.getText( charset: "UTF-8")
template=template.replaceAll("#taskName", taskName);
template=template.replaceAll("#runResultStatus", runResultStatus);
template=template.replaceAll("#status", status);
template=template.replaceAll("#message", message);
template=template.replaceAll("#progress", progress);
template=template.replaceAll("#taskOwner", taskOwner);
template=template.replaceAll("#channel", channel);
template=template.replaceAll("#handlerURI", handlerURI);
DateFormat dateFormat = new SimpleDateFormat( pattern: "dd-MMMM-yyyy");
Date date = new Date();
template=template.replaceAll( regex: "#dateShort", dateFormat.format(date));
return template
```

```
fileName = "/opt/midpoint/var/source/my-task-notification-template.html"
basic.expandTemplate(
        fileName,
        binding.variables + [dateShort:basic.formatTime(basic.now(), 'dd-MMMM-yyyy')])
```

Delta Analysis (150+ cases)

- Trying to find changes related to specific items (e.g. assignment activation) and the values being set
- Usually tens of Groovy lines because the data structures are inherently complex
- We plan to provide powerful and extensible mechanism that would contain the complexity

```
for (def delta : deltaList) {
    statusPath = ItemPath.create(UserType.F_EXTENSION, "status")
    if (delta.hasItemDelta(statusPath)) {
        propertyDelta = delta.findPropertyDelta(statusPath)
        realValue = propertyDelta?.anyValue?.value
        if (realValue == "x") {
            return "disabled"
        }
        // ...
    }
}
```

```
for (delta in deltaList) {
    if (delta.newValuesFor('extension/status').contains('x')) {
        return 'disabled'
    }
    // ...
}
```

Creating objects (250+ cases)

- Creating whole objects, assignments, inducements, references, polystrings, …
- Currently not very programmer-friendly
- Considering the builder pattern

```
AssignmentType a = new AssignmentType()
a.getSubtype().add('abc-token')
ObjectReferenceType o = new ObjectReferenceType()
o.setOid('********-****-****-****-************')
o.setType(RoleType.COMPLEX_TYPE)
a.setTargetRef(o)
```

```
basic.createRoleAssignment('********-****-****-****-************')
    .subtype('abc-token')
```

Manipulating names (150+), emails (200+), phone numbers (40+), DNs (200+ cases)

- constructing, parsing, matching, normalizing and converting these kinds of data
- e.g. person name ↔ given name, family name, title(s)
- email ↔ local part + domain
- phone # ↔ country code, area, number, extension
- DN ↔ RDNs

```
if (telephoneNumber) {
    if (telephoneNumber.startsWith('+421')) {
        return telephoneNumber.replaceFirst(/^\+421/, '0')
    } else if (telephoneNumber.length() < 10) {
        return '0' + telephoneNumber
    } else {
        return telephoneNumber
    }
}
```

```
phone.formatShort(telephoneNumber)
```

Complex Attributes

- Many resources expect and provide attributes having some internal structure
- Workarounds include passing data as JSON, XML, CSV, etc
- In 4.11 we plan to support these **complex attributes** natively, at least on a basic level

```
"{\"address\"=\""+emailAddress+"\",\"primary\"=\"true\"}"
```

Complex Attributes: An Example

```
<attribute>
    <ref>ri:address</ref> <!-- complex -->
    <inbound>
        <target>
            <path>physicalAddress</path> <!-- complex -->
        </target>
    </inbound>
</attribute>
```

```
<complexAttributeType>
    <name>ri:address</name>
    <delineation>
        <objectClass>ri:address</objectClass>
    </delineation>
    <focus>
        <type>PhysicalMailingAddressType</type>
    </focus>
    <attribute>
        <ref>ri:street</ref>
        <inbound>
            <target>
                <path>streetAddress</path>
            </target>
        </inbound>
    </attribute>
    <attribute>
        <ref>ri:city</ref>
        <inbound>
            <target>
                <path>locality</path>
            </target>
        </inbound>
    </attribute>
    <!-- ... -->
</complexAttributeType>
```

Unnecessarily complex APIs (extension items, attributes, function libraries, ...)

- `basic.getAttributeValues(shadow, [namespace], attrName)`
- `basic.getExtensionPropertyValues(object, [namespace], extItemName)`
- `myLib.execute('getAccessIdentity', [focus: focus, type: identityType])`

- `shadow.attributes.attrName`
- `object.extension.extItemName`
- `mylib.getAccessIdentity(focus, identityType)`
- `focus.getAccessIdentity(identityType)`

Querying Objects

```
def query = prismContext.queryFor(UserType)
        .item(FocusType.F_ACTIVATION, ActivationType.F_EFFECTIVE_STATUS).eq(ActivationStatusType.F_ENABLED)
        .and().item(FocusType.F_ACTIVATION, ActivationType.F_ENABLED_TIMESTAMP).gt(basic.fromNow('-P1D'))
        .build()
def result = midpoint.searchObjects(UserType, query)
```

```
def prepared = midpoint.preparedQueryFor(UserType,
        "activation matches (effectiveStatus = 'enabled' and enableTimestamp > ?)")
def query = prepared.bind(basic.fromNow("-P1D"))
def result = midpoint.searchObjects(query)
```

https://docs.evolveum.com/midpoint/reference/concepts/query/midpoint-query-language/query-language-in-groovy/

Other Areas of Improvement

- Approver, owner, manager handling (100+)
- Decision trees and mapping tables (300+)
- Pretty-printing and diagnostics (40+)
- Organization tree walking

Don’t forget to vote!
&
Send us some data :)

- Explicit business logic
- Constants
- Assignment processing
- Archetype matching
- Date/time manipulation
- Text normalization

- Notification text formatting
- Delta analysis
- Creating objects
- Manipulating names, emails, phone numbers, LDAP DNs
- Complex attributes

- Easier getting extension items and attributes, calling function libraries
- Handling owners, managers, approvers
- Decision trees and mapping tables
- Pretty-printing and diagnostics
- Organizational tree walking


Evolveum
Thank you for your attention
Feel free to ask your questions now!
Funded by the European Union
NextGenerationEU
RECOVERY AND RESILIENCE PLAN
2nd Annual
MidPoint Community Meetup