Experiments with OntoUML Catalog

In [1]:
```
import os
import glob
import json
import pandas as pd
import numpy as np
```

In [2]:
```
import requests
```

In [3]:
```
import textwrap
from pandas.api.types import CategoricalDtype
```

In [4]:
```
import matplotlib.pyplot as plt
import matplotlib as mpl
import seaborn as sns
%matplotlib inline
```

In [5]:
```
dir_implement = os.getcwd()
os.chdir("Images")
dir_images = os.getcwd()
os.chdir("../Abstractions")
dir_abstractions = os.getcwd()
os.chdir("../Errors")
dir_errors = os.getcwd()
os.chdir("../../../GitHub/ontouml-models2/models")
dir_models = os.getcwd()
```

In [6]:
```
sns.set_theme(style="whitegrid", palette="pastel")
sns.despine(offset=5, trim=True)
```

```
<Figure size 432x288 with 0 Axes>
```

In [7]:
```
colors={
    'super small': 'magenta',
    'small': 'green',
    'medium': 'blue',
    'big': 'orange',
    'super big': 'indigo'
}
```

Preparing subsets of models

List of potential models

Setting a directory with models as a working directory...

In [8]:
```
os.chdir(dir_models)
os.getcwd()
```

...

In [9]:
```
json_problems = [
    'digitaldoctor2022/ontology.json',
    'goncalves2011ecg/ontology.json',
    'tourbo2021/ontology.json',
    'plato-ontology2019/ontology.json',
    'buridan-ontology2021/ontology.json',
    'aristotle-ontology2019/ontology.json',
    'public-expense-ontology2020/ontology.json',
    'tender2013/ontology.json',
    'scientific-publication2013/ontology.json'
]
```

Go to the folder with models and scan it for *.json

In [10]:
```
all_file_names = []
for file in glob.glob("*/ontology.json"):
    if file not in json_problems:
        all_file_names.append(file)

print(f"We have {len(all_file_names)} files with ontologies.")
```

```
We have 159 files with ontologies.
```

In order to select only those models, that contains only 16 stereotypes (those, for which the algorithm was developed), we

1. analyse all models
2. filter those of our interest

In [11]:
```
def normalize(stereotype: str) -> str:
    if stereotype:
        stereotype = stereotype.lower().replace(" ", "")
    return stereotype
```

In [12]:
```
def get_all_stereotypes(contents, all_content) -> dict:
    if contents:
        for content in contents:
            if content['type'] == 'Package':
                all_content = get_all_stereotypes(content['contents
            else:
                if content['type'] == 'Class':
                    if 'stereotype' in content.keys():
                        stereotype = content['stereotype']
                        stereotype = normalize(stereotype)
                        #if stereotype:
                        if stereotype in all_content:
                            all_content[stereotype] += 1
                        else:
                            all_content[stereotype] = 1
                    elif 'stereotypes' in content.keys():
                        if content['stereotypes']:
                            for stereotype in content['stereotypes'
                                if stereotype in all_content:
                                    all_content[stereotype] += 1
                                else:
                                    all_content[stereotype] = 1
    return all_content
```

In [13]:
```
df_stereotypes = pd.DataFrame(columns=['Name'])

for file_name in all_file_names:
    file = open(file_name, encoding="ISO-8859-1", mode="r")
    data = json.loads(file.read())
    if 'model' in data.keys():
        contents = data['model']['contents']
        model_stereotypes = get_all_stereotypes(contents, {})
        model_stereotypes['Name'] = file_name.split('/')[0]
        df_stereotypes = df_stereotypes.append(model_stereotypes, i
    else:
        print(f"ERROR: Model not found in {file_name}.")
    file.close()

df_stereotypes = df_stereotypes.fillna(0)
df_stereotypes = df_stereotypes.set_index('Name')
df_stereotypes = df_stereotypes.astype(int)

print(f"We have stereotypes for {len(df_stereotypes)} ontologies.")
```

```
We have stereotypes for 159 ontologies.
```

In [14]:
```
df_stereotypes.head()
```

Out[14]:

Name	category	event	kind	mode	relator	role	rolemixin	collective	subk
falduci2022non-consensual-pornography	3	3	2	4	4	9	2	0	
clergy-ontology	0	0	1	0	7	16	0	5	
elikan2018brand-identity	3	0	10	8	1	0	0	6	
services2015	4	0	5	8	5	0	10	1	
pereira2020ontotrans	0	0	0	0	0	0	35	0	

5 rows × 64 columns

Just for curiosity, what are the most popular stereotypes?

In [15]:
```
df_stereotypes.sum().sort_values(ascending=False)[0:10]
```

Out[15]:
```
role          2074
subkind       2030
kind          1657
relator       1327
goal           952
NaN            786
category       588
rolemixin      559
mode           525
event          432
dtype: int64
```

Filtering only those models, that can be processed by the algorithm

In [16]:
```
algorithm_stereotypes = [
    'subkind', 'kind', 'role', 'relator', 'category',
    'event', 'rolemixin', 'mode', 'phase', 'collective',
    'datatype', 'quality', 'mixin', 'quantity',
    'enumeration', 'phasemixin'
]
```

In [17]:
```
subset = df_stereotypes.columns.difference(algorithm_stereotypes)
not_supported_models = df_stereotypes[df_stereotypes[subset].sum(ax
print(f"Number of models that contains not supported class stereoty

df_models = df_stereotypes.loc[~df_stereotypes.index.isin(not_suppo
                               ~df_stereotypes.columns.isin(subset)
```

```
Number of models that contains not supported class stereotypes: 72
```

In [18]:
```
potential_file_names = [name for name in all_file_names if name.spl
print(f"Number of models that can be processed: {len(potential_file
```

```
Number of models that can be processed: 87
```

In [19]:
```
def get_content(contents, all_content) -> dict:
    if contents:
        for content in contents:
            if content['type'] == 'Package':
                all_content = get_content(content['contents'], all_
            else:
                if content['type'] == 'Class':
                    all_content['Classes'] += 1
                elif content['type'] == 'Relation':
                    if (content['properties'][0]['aggregationKind']
                        content['properties'][1]['aggregationKind']
                        all_content['PartOf'] += 1
                    all_content['Relations'] += 1
                elif content['type'] == 'Generalization':
                    all_content['Generalizations'] += 1
                    all_content['Relations'] += 1
    return all_content
```

In [20]:
```
df_potential = pd.DataFrame(columns=['Name', 'Classes', 'Relations'

for file_name in potential_file_names:
    file = open(file_name, encoding="ISO-8859-1", mode="r")
    data = json.loads(file.read())
    contents = None
    if 'contents' in data.keys():
        contents = data['contents']
    elif 'model' in data.keys():
        contents = data['model']['contents']
    else:
        print(f"ERROR: Neither model nor contents found in {file_na
    file.close()

    all_content = get_content(contents,
                              {
                                  'Classes': 0,
                                  'Relations': 0,
                                  'PartOf': 0,
                                  'Generalizations': 0
                              })
    all_content['Name'] = file_name.split('/')[0]
    df_potential = df_potential.append(all_content, ignore_index =

df_potential = df_potential.fillna(0)
df_potential = df_potential.set_index('Name')
df_potential = df_potential.astype(int)

print(f"We have statistics for {len(df_potential)} models.")
```

```
We have statistics for 87 models.
```

In [21]:
```
df_potential.describe()
```

Out[21]:

	Classes	Relations	Generalizations	PartOf
count	87.000000	87.000000	87.000000	87.000000
mean	38.137931	55.011494	26.091954	4.908046
std	54.410407	102.732606	66.152156	10.581502
min	7.000000	9.000000	0.000000	0.000000
25%	18.000000	26.000000	9.000000	0.000000
50%	29.000000	37.000000	15.000000	1.000000
75%	41.500000	59.500000	24.000000	5.000000
max	487.000000	956.000000	608.000000	84.000000

In [22]:
```
df_potential['TotalSize'] = df_potential['Classes'] + df_potential[
print(df_potential['TotalSize'].sort_values(ascending=False)[0:10])
```

```
Name
barcelos2015transport-networks    1443
kritz2020ontobg                    399
xhani2023xmlpo                     251
freshbz2023                        210
silva2012itarchitecture            187
eu-rent-refactored2022             169
public-tender                      154
internal-affairs2013               150
aguiar2019ooco                     133
dpo2017                            126
Name: TotalSize, dtype: int64
```

In [23]:
```
conditions = [
    (df_potential['TotalSize'] >= 1000),
    (df_potential['TotalSize'] < 1000) & (df_potential['TotalSize']
    (df_potential['TotalSize'] < 200) & (df_potential['TotalSize']
    (df_potential['TotalSize'] < 75) & (df_potential['TotalSize'] >
    (df_potential['TotalSize'] < 35)
]
values = ['super big', 'big', 'medium', 'small', 'super small']
df_potential['Model size'] = np.select(conditions, values)
df_potential.head()
```

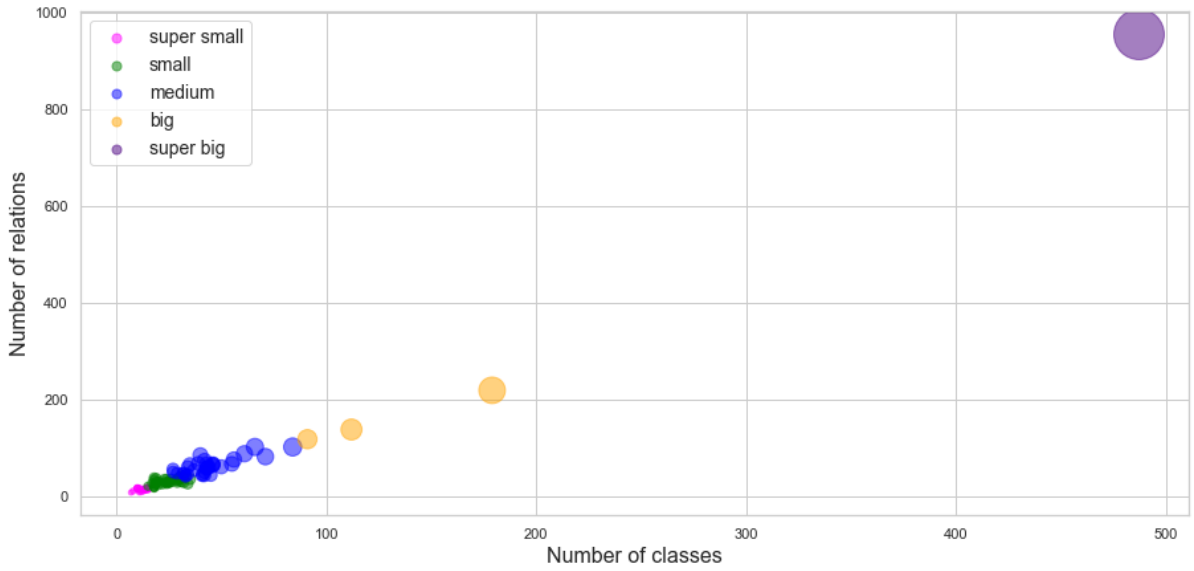
Out[23]:

Name	Classes	Relations	Generalizations	PartOf	TotalSize	Model size
falduci2022non-consensual-pornography	27	51	13	4	78	medium
clergy-ontology	29	49	16	6	78	medium
elikan2018brand-identity	30	37	6	0	67	small
pereira2020ontotrans	35	67	0	0	102	medium
barcelos2013normative-acts	45	45	27	18	90	medium

In [24]:
```
os.chdir(dir_images)
os.getcwd()
```

...

In [25]:
```
fig, ax = plt.subplots()
fig.set_figwidth(15)
fig.set_figheight(7)

for (t,c) in colors.items():
    sel_df = df_potential[df_potential['Model size']==t]
    scatter = ax.scatter(sel_df['Classes'], sel_df['Relations'], s=
                         alpha=0.5, c=c, cmap='viridis', label=t)

#plt.title("Size of conceptual models", fontsize=16)
plt.xlabel("Number of classes", fontsize=16)
plt.ylabel("Number of relations", fontsize=16)

lgnd = plt.legend(markerscale=1,scatterpoints=1, fontsize=14)

#change the marker size manually for all
lgnd.legendHandles[0]._sizes = [50]
lgnd.legendHandles[1]._sizes = [50]
lgnd.legendHandles[2]._sizes = [50]
lgnd.legendHandles[3]._sizes = [50]
lgnd.legendHandles[4]._sizes = [50]
#plt.show()
plt.savefig('all_models.png')
```



List of valid models

Send request to `api.ontouml.org` and check models for validity.

In [26]:
```
headers = {
    'Accept': "application/json",
    'Connection': "keep-alive"
}
```

In [27]:
```
url_verify = "http://api.ontouml.org/v1/verify"
```

List of fixed models:

1. bernasconi2023fair-principles ontology 1 errors were found
2. In goncalves2011ecg ontology 1 errors were found
3. In gomes2022digital-technology ontology 1 errors were found
4. In eu-rent-refactored2022 ontology 2 errors were found
5. In health-organizations ontology 5 errors were found
6. In srro-ontology ontology 2 errors were found
7. In aguiar2019ooco ontology 3 errors were found
8. In nardi2015ufo-s ontology 1 errors were found

In [28]:
```
os.chdir(dir_models)
os.getcwd()
```

...

In [29]:
```
valid_file_names = []

for file_name in potential_file_names:
    file = open(file_name, encoding="ISO-8859-1", mode="r")
    data = json.loads(file.read())
    file.close()

    body = {'project': data}
    response = requests.post(url_verify, headers=headers, json=body
    responseResults = json.loads(response.text)['result']
    if len(responseResults) == 0:
        valid_file_names.append(file_name)
    else:
        print(f"In {file_name.split('/')[0]} ontology {len(response

print(f"Number of valid ontologies is {len(valid_file_names)}")
```

```
In falduci2022non-consensual-pornography ontology 13 errors were f
ound
In elikan2018brand-identity ontology 4 errors were found
In ramirez2015userfeedback ontology 17 errors were found
In castro2012cloudvulnerability ontology 5 errors were found
In public-tender ontology 21 errors were found
In road-accident2013 ontology 1 errors were found
In laurier2018rea ontology 8 errors were found
In idaf2013 ontology 1 errors were found
In guarino2018rea ontology 6 errors were found
In zanetti2019orm-o ontology 20 errors were found
In bank-model ontology 4 errors were found
In chartered-service ontology 8 errors were found
In music-ontology ontology 8 errors were found
In photography ontology 4 errors were found
In dpo2017 ontology 14 errors were found
In library ontology 14 errors were found
In silveira2021oap ontology 17 errors were found
In zhou2017hazard-ontology-robotic-strolling ontology 12 errors we
re found
In public-organization2013 ontology 2 errors were found
In short-examples2013 ontology 2 errors were found
```

```
In heirbrant2023boekbazaar ontology 6 errors were found
In barcelos2015transport-networks ontology 43 errors were found
In pereira2015doacao-orgaos ontology 8 errors were found
In project-assets2013 ontology 2 errors were found
In santos2020valuenetworks ontology 18 errors were found
In haridy2021egyptian-e-government ontology 4 errors were found
In aguiar2018rdbs-o ontology 13 errors were found
In kritz2020ontobg ontology 72 errors were found
In internship ontology 7 errors were found
In zhou2017hazard-ontology-train-control ontology 20 errors were f
ound
In guizzardi2005ontological ontology 3 errors were found
In junior2018o4c ontology 5 errors were found
In bank-account2013 ontology 6 errors were found
In fischer2018ontorea ontology 18 errors were found
In rocha2023ciencia-aberta ontology 3 errors were found

In university-ontology ontology 10 errors were found
In elghosh2020cargos ontology 8 errors were found
In machacova2023gym ontology 2 errors were found
Number of valid ontologies is 49
```

In [30]:
```
print(f"Number of valid ontologies is {len(valid_file_names)}")
print(f"Number of potential ontologies is {len(potential_file_names
```

```
Number of valid ontologies is 49
Number of potential ontologies is 87
```

In [31]:
```
df_valid = df_potential.loc[df_potential.index.isin([name.split('/'
df_valid.head()
```

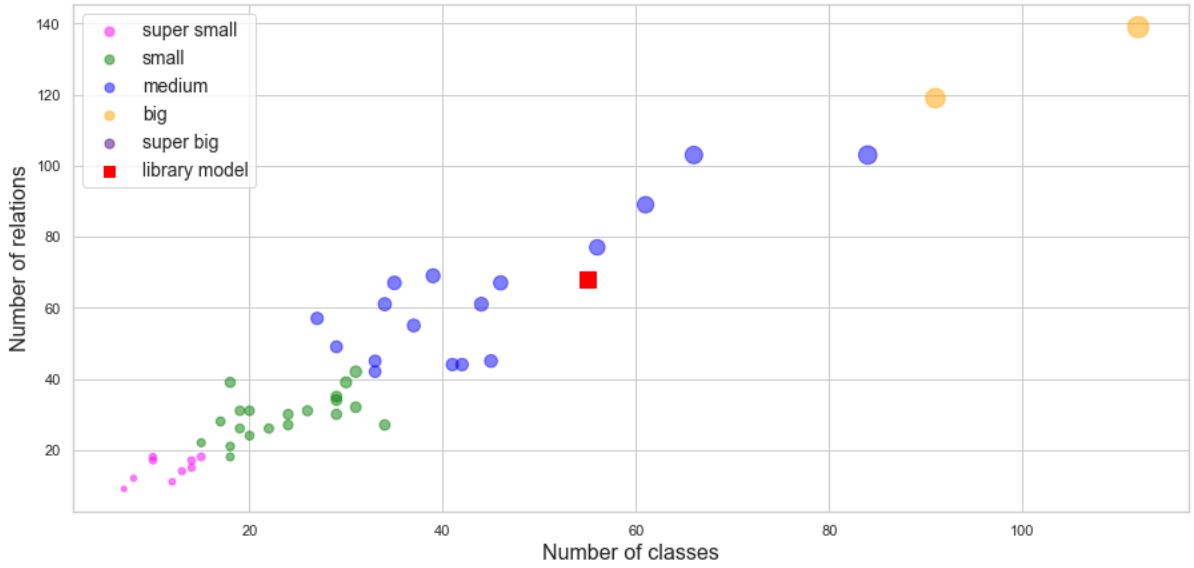
Out[31]:

Name	Classes	Relations	Generalizations	PartOf	TotalSize	Model size
clergy-ontology	29	49	16	6	78	medium
pereira2020ontotrans	35	67	0	0	102	medium
barcelos2013normative-acts	45	45	27	18	90	medium
buchtela2020connection	19	26	13	0	45	small
martinez2013human-genome	10	18	3	4	28	super small

In [33]:
```
os.chdir(dir_images)
os.getcwd()
```

...

In [34]:
```
fig, ax = plt.subplots()
fig.set_figwidth(15)
fig.set_figheight(7)

library = df_valid.loc['romanenko2023what']
for (t,c) in colors.items():
    sel_df = df_valid[df_valid['Model size']==t]
    scatter = ax.scatter(sel_df['Classes'], sel_df['Relations'], s=
                         alpha=0.5, c=c, cmap='viridis', label=t)

ax.scatter(library['Classes'], library['Relations'], s=library['Tot
           alpha=1.0, c='red', marker='s', cmap='viridis', label='

#plt.title("Size of conceptual models", fontsize=16)
plt.xlabel("Number of classes", fontsize=16)
plt.ylabel("Number of relations", fontsize=16)

lgnd = plt.legend(markerscale=1,scatterpoints=1, fontsize=14)

#change the marker size manually for all
lgnd.legendHandles[0]._sizes = [50]
lgnd.legendHandles[1]._sizes = [50]
lgnd.legendHandles[2]._sizes = [50]
lgnd.legendHandles[3]._sizes = [50]
lgnd.legendHandles[4]._sizes = [50]
lgnd.legendHandles[5]._sizes = [50]

plt.savefig('valid_models.png')
```



Running abstractions on different sets

Checking valid models

In [35]:
```
os.chdir(dir_models)
os.getcwd()
```

...

In [36]:
```
url_abstract = "https://expose.eng.unibz.it/abstract"
```

In [37]:
```
atypes = {
    "h": ['hierarchy'],
    "a": ['aspects'],
    "p": ['parthood'],
    "ha": ['hierarchy', 'aspects'],
    "ap": ['parthood', 'aspects'],
    "hp": ['parthood', 'hierarchy'],
    "full": ['parthood', 'hierarchy', 'aspects']
}
```

In [38]:
```
%%time
for file_name in valid_file_names:
    model_name = file_name.split(os.path.sep)[0]
    file = open(file_name, encoding="ISO-8859-1", mode="r")
    data = json.loads(file.read())

    for abstr_name, abstr_params in atypes.items():
        response = requests.post(url_abstract, headers=headers,
                                 json={
                                     'abs_type': abstr_params,
                                     'long_names': True,
                                     'mult_relations': False,
                                     'keep_relators': True,
                                     'in_format': 'json',
                                     'out_format': 'json',
                                     'height': 1000,
                                     'width': 1000,
                                     'origin': data
                                    #'origin': json.load(open(file
                                 })

        if response.ok:
            new_file_name = f"{dir_abstractions}{os.path.sep}{model
            with open(new_file_name, 'w') as f:
                json.dump(response.json(), f)

print(f"All valid models were processed.")
```

```
All valid models were processed.
CPU times: user 10.1 s, sys: 1.11 s, total: 11.2 s
Wall time: 2min 13s
```

In [39]:
```
os.chdir(dir_abstractions)
```

In [40]:
```
abstraction_file_names = []
for file in glob.glob("*.json"):
    abstraction_file_names.append(file)

print(f"We have {len(abstraction_file_names)} files with abstractio
```

```
We have 343 files with abstractions.
```

Validation check of abstracted models

In [41]:
```
%%time
df_abstract = pd.DataFrame(columns=['Name', 'Classes', 'Relations',

for file_name in abstraction_file_names:
    file = open(file_name, encoding="ISO-8859-1", mode="r")
    data = json.loads(file.read())
    contents = None
    if 'contents' in data.keys():
        contents = data['contents']
    elif 'model' in data.keys():
        contents = data['model']['contents']
    else:
        print(f"ERROR: Neither model nor contents found in {file_na
    file.close()

    all_content = get_content(contents,
                              {
                                  'Classes': 0,
                                  'Relations': 0,
                                  'PartOf': 0,
                                  'Generalizations': 0
                              })
    all_content['Name'] = file_name.split('/')[0][:-5]

    response = requests.post(url_verify, headers=headers, json={'pr
    responseResults = json.loads(response.text)
    if 'result' not in responseResults:
        print(all_content['Name'] + ": " + responseResults['message
    df_abstract = df_abstract.append(all_content, ignore_index = Tr

df_abstract = df_abstract.fillna(0)

print(f"We have statistics for {len(df_abstract)} abstractions.")
```

```
aguiar2019ooco_ap: The input could not be parse into a valid insta
nce of Project.
aguiar2019ooco_hp: The input could not be parse into a valid insta
nce of Project.
aguiar2019ooco_full: The input could not be parse into a valid ins
tance of Project.
We have statistics for 343 abstractions.
CPU times: user 3.98 s, sys: 447 ms, total: 4.43 s
Wall time: 1min 1s
```

In [42]:
```
anames = {
    "h": 'hierarchy',
    "a": 'aspects',
    "p": 'parthood',
    "ha": 'aspects and hierarchy',
    "ap": 'parthood and aspects',
    "hp": 'parthood and hierarchy',
    "full": 'full abstraction'
}
```

In [43]:
```
df_abstract["TotalSize"] = df_abstract["Classes"] + df_abstract["Re
df_abstract['Model size'] = ""
df_abstract["Type of abstraction"] = df_abstract["Name"].str.rsplit
df_abstract["Name"] = df_abstract["Name"].str.rsplit('_', 1).str[0]
size_dict = pd.Series(df_valid['Model size'].values,index=df_valid.
df_abstract['Model size'] = df_abstract['Name'].map(size_dict)
```

In [44]:
```
df_abstract.head()
```

Out[44]:

	Name	Classes	Relations	Generalizations	PartOf	TotalSize	Model size	Type abstracti
0	xhani2023xmlpo	59	75	0	0	134	big	partho a hierarc
1	genealogy2013	1	1	0	0	2	super small	aspe a hierarc
2	stock-broker2021	13	16	11	0	29	super small	aspe
3	genealogy2013	7	9	8	0	16	super small	partho a aspe
4	telecom-equipment2013	18	31	0	0	49	medium	hierarc

In [45]:
```
original_models = df_valid.copy(deep=True).reset_index()
original_models["Type of abstraction"] = 'original model'
original_models.head()
```

Out[45]:

	Name	Classes	Relations	Generalizations	PartOf	TotalSize	Model size	a
0	clergy-ontology	29	49	16	6	78	medium	
1	pereira2020ontotrans	35	67	0	0	102	medium	
2	barcelos2013normative-acts	45	45	27	18	90	medium	
3	buchtela2020connection	19	26	13	0	45	small	
4	martinez2013human-genome	10	18	3	4	28	super small	

In [46]:
```
df_abstract = pd.concat([df_abstract,original_models], ignore_index
```

In [47]:
```
df_abstract[df_abstract['Name']=='romanenko2023what']
```

Out[47]:

	Name	Classes	Relations	Generalizations	PartOf	TotalSize	Model size	ab
7	romanenko2023what	36	41	26	0	77	medium	
12	romanenko2023what	16	22	0	1	38	medium	
120	romanenko2023what	26	38	0	1	64	medium	
168	romanenko2023what	53	63	35	0	116	medium	
215	romanenko2023what	24	34	0	0	58	medium	
310	romanenko2023what	38	46	28	3	84	medium	
311	romanenko2023what	14	18	0	0	32	medium	a
351	romanenko2023what	55	68	37	3	123	medium	

In [48]:
```
abs_types = ['original model', 'aspects', 'parthood',
             'parthood and aspects', 'hierarchy',
             'aspects and hierarchy', 'parthood and hierarchy',
             'full abstraction']
abstraction_type = CategoricalDtype(abs_types, ordered=True)
df_abstract['Type of abstraction'] = df_abstract['Type of abstracti
df_abstract.sort_values(by='Type of abstraction', inplace=True)
```

In [49]:
```
df_abstract.head(10)
```

Out[49]:

	Name	Classes	Relations	Generalizations	PartOf	TotalSize	Model size
391	fumagalli2022criminal-investigation	10	17	8	0	27	super small
364	sousa2022triponto	44	61	25	1	105	medium
378	qam	41	44	24	0	85	medium
377	srro-ontology	20	31	16	3	51	small
376	health-organizations	24	27	14	0	51	small
360	formula-one2023	26	31	18	4	57	small
345	barcelos2013normative-acts	45	45	27	18	90	medium
344	pereira2020ontotrans	35	67	0	0	102	medium
375	fernandez-cejas2022curie-o	29	35	14	3	64	small
343	clergy-ontology	29	49	16	6	78	medium

In [50]:
```
os.chdir(dir_images)
os.getcwd()
```

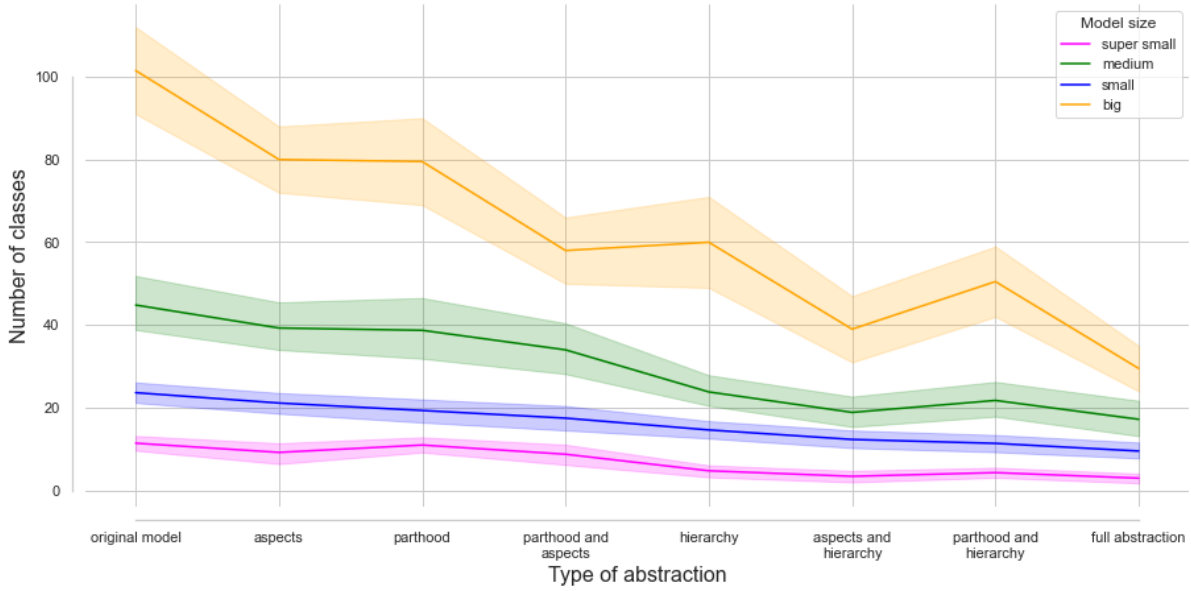
...

In [51]:
```
df_abstract.to_excel("output.xlsx", sheet_name='abstractions')
```

In [52]:
```
def wrap_labels(ax, width, break_long_words=False):
    labels = []
    for label in ax.get_xticklabels():
        text = label.get_text()
        labels.append(textwrap.fill(text, width=width,
                      break_long_words=break_long_words))
    ax.set_xticklabels(labels, rotation=0)
```

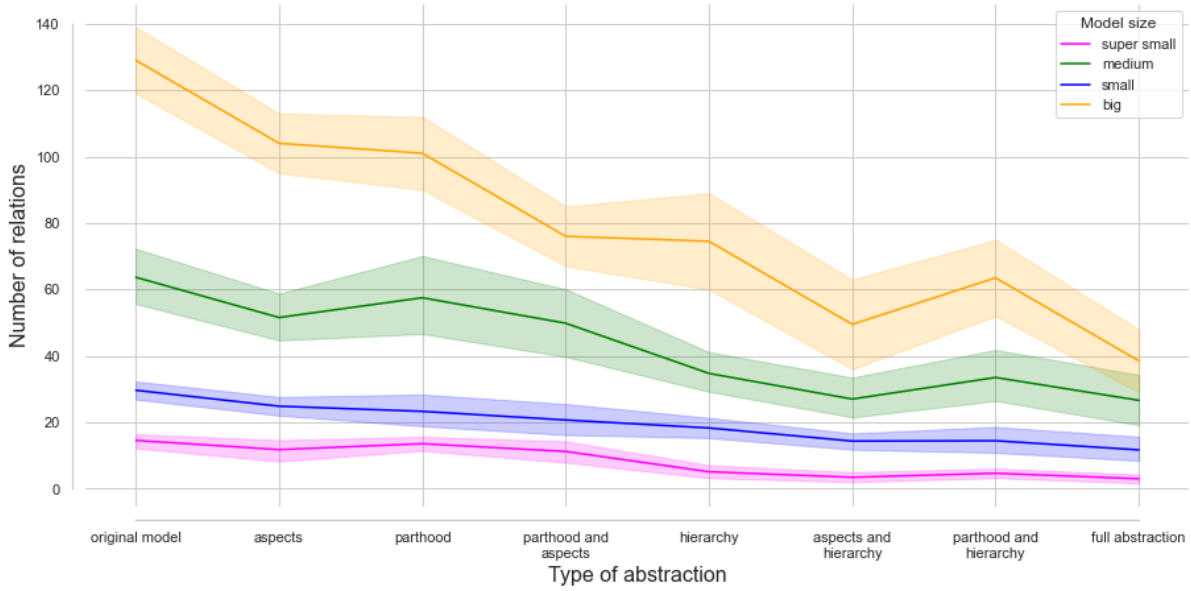
In [53]:
```
fig, ax = plt.subplots(figsize=(15, 7))
sns.lineplot(x="Type of abstraction", y="Classes",
             hue="Model size", palette=colors.values(),
             data=df_abstract, ax=ax, sort=False)
ax.set_xticklabels(abs_types)
wrap_labels(ax, 18)
sns.despine(offset=10, trim=True)
plt.xlabel('Type of abstraction', fontsize=16)
plt.ylabel('Number of classes', fontsize=16)
#plt.show()
plt.savefig('classes_compression.png')
```

```
<ipython-input-53-d6b75df396b8>:5: UserWarning: FixedFormatter sho
uld only be used together with FixedLocator
  ax.set_xticklabels(abs_types)
<ipython-input-52-defba7c600e5>:7: UserWarning: FixedFormatter sho
uld only be used together with FixedLocator
  ax.set_xticklabels(labels, rotation=0)
```



In [54]:
```
fig, ax = plt.subplots(figsize=(15, 7))
sns.lineplot(x="Type of abstraction", y="Relations",
             hue="Model size", palette=colors.values(),
             data=df_abstract, ax=ax, sort=False)
ax.set_xticklabels(abs_types)
wrap_labels(ax, 18)
sns.despine(offset=10, trim=True)
plt.xlabel('Type of abstraction', fontsize=16);
plt.ylabel('Number of relations', fontsize=16);
#plt.show()
plt.savefig('relations_compression.png')
```

```
<ipython-input-54-2acb70261d14>:5: UserWarning: FixedFormatter sho
uld only be used together with FixedLocator
  ax.set_xticklabels(abs_types)
<ipython-input-52-defba7c600e5>:7: UserWarning: FixedFormatter sho
uld only be used together with FixedLocator
  ax.set_xticklabels(labels, rotation=0)
```



Checking potential models

In [68]:
```
os.chdir(dir_models)
os.getcwd()
```

...

In [69]:
```
error_file_names = list(set(potential_file_names) - set(valid_file_
len(error_file_names)
```

Out[69]: 38

In [70]:
```
%%time
for file_name in error_file_names:
    model_name = file_name.split(os.path.sep)[0]

    for abstr_name, abstr_params in atypes.items():
        file = open(file_name, encoding="ISO-8859-1", mode="r")
        data = json.loads(file.read())
        response = requests.post(url_abstract, headers=headers,
                                 json={
                                     'abs_type': abstr_params,
                                     'long_names': True,
                                     'mult_relations': False,
                                     'keep_relators': True,
                                     'in_format': 'json',
                                     'out_format': 'json',
                                     'height': 1000,
                                     'width': 1000,
                                     'origin': data
                                 })

        if response.ok:
            new_file_name = f"{dir_errors}{os.path.sep}{model_name}
            with open(new_file_name, 'w') as f:
                json.dump(response.json(), f)

print(f"All models with errors were processed.")
```

```
All models with errors were processed.
CPU times: user 8.96 s, sys: 969 ms, total: 9.93 s
Wall time: 1min 45s
```

In [71]:
```
df_error = pd.DataFrame(columns=['Name'] + abs_types)
```

In [72]:
```
for file_name in error_file_names:
    file = open(file_name, encoding="ISO-8859-1", mode="r")
    data = json.loads(file.read())
    file.close()

    body = {'project': data}
    response = requests.post(url_verify, headers=headers, json=body
    responseResults = json.loads(response.text)['result']
    df_error = df_error.append({'Name':file_name.split('/')[0],
                                'original model': len(responseResul
                               ignore_index = True)
df_error = df_error.set_index('Name')
```

In [73]:
```
os.chdir(dir_errors)
```

In [74]:
```
abstraction_error_file_names = []
for file in glob.glob("*.json"):
    abstraction_error_file_names.append(file)

print(f"We have {len(abstraction_error_file_names)} files with abst
```

```
We have 259 files with abstractions.
```

In [75]:
```
%%time
for file_name in abstraction_error_file_names:
    file = open(file_name, encoding="ISO-8859-1", mode="r")
    data = json.loads(file.read())
    response = requests.post(url_verify, headers=headers, json={'pr
    responseResults = json.loads(response.text)
    if 'result' not in responseResults:
        print(file_name + ": " + responseResults['message'])
    else:
        name, abstraction = file_name.rsplit('_', 1)
        df_error.loc[name, anames[abstraction[:-5]]] = len(response
df_error = df_error.fillna(0)
df_error = df_error.astype(int)
```

```
CPU times: user 2.25 s, sys: 413 ms, total: 2.66 s
Wall time: 48.3 s
```

In [83]:
```
df_error.describe()
```

Out[83]:

	original model	aspects	parthood	parthood and aspects	hierarchy	aspects and hierarchy	parthood and hierarchy	abstra
count	38.000000	38.000000	38.000000	38.000000	38.000000	38.000000	38.000000	38.0(
mean	11.289474	8.842105	9.526316	8.131579	3.526316	3.526316	3.526316	3.5
std	13.005005	12.119955	11.103291	11.194041	4.642558	4.642558	4.642558	4.6
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0(
25%	4.000000	2.000000	4.000000	2.000000	0.000000	0.000000	0.000000	0.0(
50%	8.000000	6.000000	6.000000	5.000000	2.000000	2.000000	2.000000	2.0(
75%	14.000000	12.750000	12.750000	11.500000	4.000000	4.000000	4.000000	4.0(
max	72.000000	72.000000	65.000000	65.000000	20.000000	20.000000	20.000000	20.0(

Complete check of models

In [85]:
```
os.chdir(dir_models)
os.getcwd()
```

...

In [88]:
```
%%time
all_models = 0
for idx, file_name in enumerate(valid_file_names):
    model_name = file_name.split(os.path.sep)[0]
    print(f"({idx}) {model_name}")

    file = open(file_name, encoding="ISO-8859-1", mode="r")
    data = json.loads(file.read())
    rule = "start"
    applied_rules = []

    while rule:
        response = requests.post(url_abstract, headers=headers,
                                 json={
                                     'abs_type': [],
                                     'long_names': True,
                                     'mult_relations': False,
                                     'keep_relators': True,
                                     'in_format': 'json',
                                     'out_format': 'expo',
                                     'height': 1000,
                                     'width': 1000,
                                     'origin': data
                                 })
        if response.ok:
            all_models += 1
            abstraction = json.loads(response.text)
            rule = abstraction["rule"]
            applied_rules.append(rule)
            data = abstraction["origin"]

            response = requests.post(url_verify, headers=headers, j
            responseResults = json.loads(response.text)
            if 'result' not in responseResults:
                print(f"ERROR: Not valid abstraction of {model_name
                print(responseResults['message'])
                # rule = ""
                # break

        else:
            print(f"ERROR: Cannot abstract model {model_name} at st
            rule = ""
            break

    print(", ".join(applied_rules)[:-2])
print(f"Total number of all models is {all_models}")
```

...

In [ ]: