Amandine Lee,
PyCon 2017
Portland, OR

Actually write speaker notes or notecards if I won't able to see them live.

Passing Exceptions 101

PARADIGMS IN ERROR HANDLING

CONTEXT AS INTRODUCTION

Recurse Center

Dropbox

Release Engineering

write bullet points or scripts for this

QUESTIONS I WANTED TO ANSWER

- How do make my code correct and reliable?
- But also readable and maintainable?

TOPICS NOT COVERED

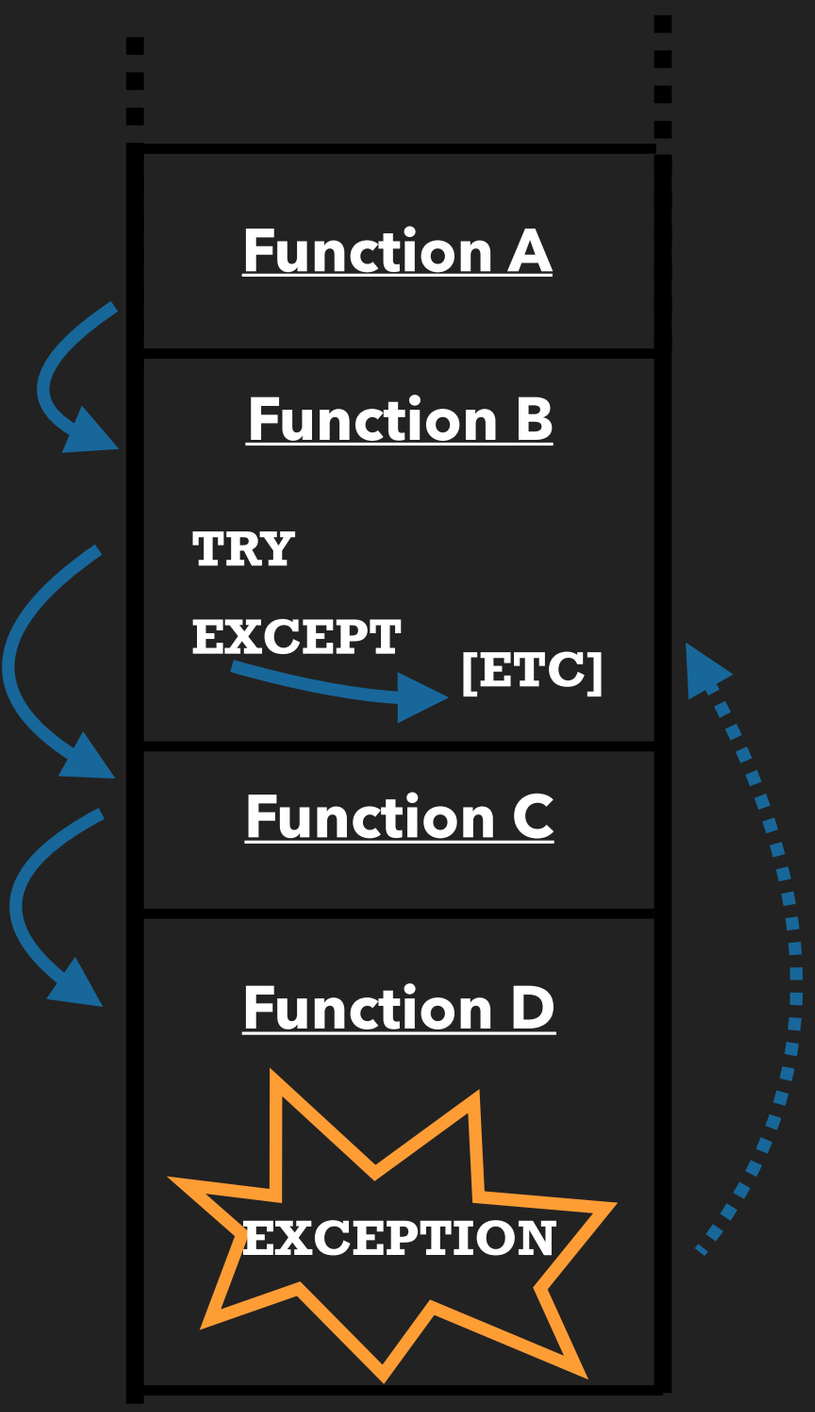
- Underlying implementation of exceptions
- Performance

Tools of the trade

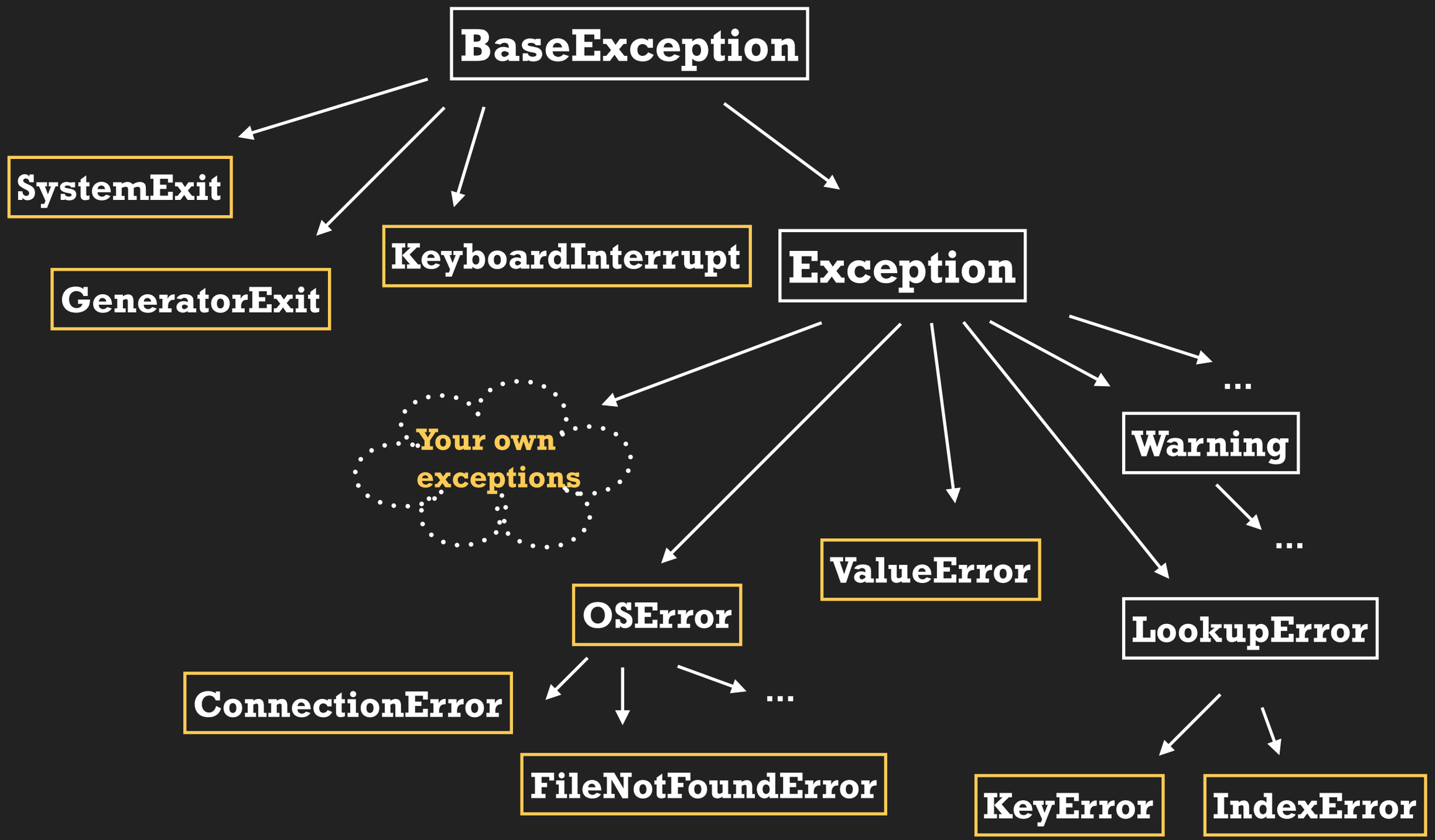
WHAT IS AN EXCEPTION?

EXCEPTION USAGE

```
1  try:
2      do_thing()
3  except OSError:
4      retry()
5  except Exception as exc:
6      crash_dump(exc)
7      raise exc
8
9  class MyError(Exception):
10     pass
11
12 raise MyError()
```



BUILT IN EXCEPTIONS (PYTHON 3.6)



Let's start digging

AN EXAMPLE

FIRST DRAFT

```
1 CONFIG_FILE = 'config.json'
2
3 def get_id(conf_fname=CONFIG_FILE):
4     with open(conf_fname, 'r') as fobj:
5         data = json.load(fobj)
6     try:
7         return data['id']
8     except KeyError:
9         return None
```

REASONS FOR EXCEPTIONS

1) Something exceptional happened

ENUMERATING POSSIBLE ERRORS

```
def get_id(conf_fname=CONFIG_FILE):
    try:
        with open(conf_fname, 'r') as fobj:
            data = json.load(fobj)
    except FileNotFoundError, JSONDecodeError:
        raise MalformedDataError()

try:
    return data['id']
except TypeError:
    raise MalformedDataError()
except KeyError:
    return None
```

REASONS FOR EXCEPTIONS

1) Something exceptional happened

2) Something went wrong

3) Something went wrong

CATCH 'EM ALL

```
1 def get_id(conf_fname=CONFIG_FILE):
2     try:
3         with open(CONFIG_FILE, 'r') as fobj:
4             data = json.loads(fobj)
5         return data.get('id')
6     except Exception:
7         raise MalformedDataError()
```

REASONS FOR EXCEPTIONS

1) Control flow

2) Unexpected *external* error

3) Bug in my code

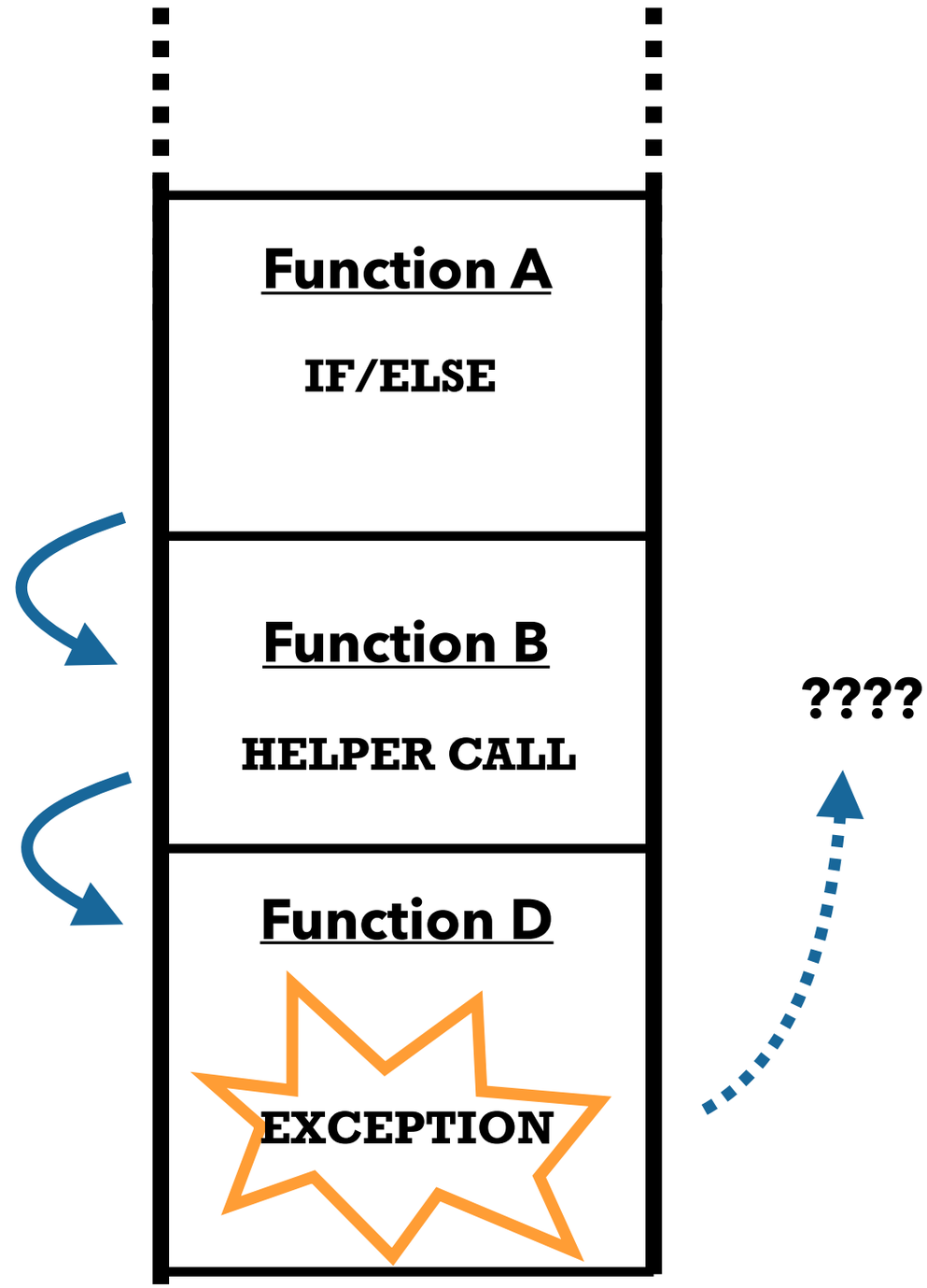
CAN WE BE MORE SYSTEMATIC?

Architecture patterns

EXCEPTION USE CASES

1. CONTROL FLOW

- Do you expect this exception?
- Do you *need* to jump the stack?
- Can you avoid provoking control flow exceptions?



3. BUG IN MY CODE

```
>>> deg get_id():
  File "<stdin>", line 1
    deg get_id():
               ^
SyntaxError: invalid syntax
```

Raise errors early

- Mypy! Static analysis!

Raise ‘em explicitly where they came from

- Don’t hide with general try/catch
- Communicate the context

2. UNEXPECTED EXTERNAL ERROR

Assigning responsibility

DESIGN BY CONTRACT

HOW CONTRACTS WORK

Function Contract

Client:

- Supply pre-co

Supplier:

- Once pre-conditions are met, fulfill post-conditions

Everyone:

- Maintain invariants

Replace with a diagram?
delegation and agreements?

CONTRACT ELEMENTS

Input types/values

Outputs types/value

Error and exceptions raised

Side effects

Preconditions

Postconditions

Invariants

```
class interface
    ACCOUNT

feature -- Element change

    deposit (sum: INTEGER)
            -- Add `sum' to account.
        require
            non_negative: sum >= 0
        ensure
            one_more_deposit:
                deposit_count =
                old deposit_count + 1
            updated: balance =
                 old balance + sum

invariant
    consistent_balance: balance =
        all_deposits.total

end -- class interface ACCOUNT
```

What exactly do these keywords DO?

https://www.eiffel.org/doc/eiffel/ET%3A%20Design%20by%20Contract%20%28tm%29%2C%20Assertions%20and%20Exceptions

WHO ENFORCES THE CONTRACT?

```
 1 def get_id(conf_fname=CONFIG_FILE):
 2     """Return the user id from CONFIG_FILE, or None if it hasn't been set
yet"""
 3     if not isinstance(conf_fname, str):
 4         raise TypeError('conf_fname must be a string')
 5     if not (os.path.exists(conf_fname) and os.path.isfile(conf_fname):
 6         raise ValueError('Invalid value for config filename: %r' % conf_fname)
 7     try:
 8         with open(conf_fname, 'r') as fobj:
 9             data = json.loads(fobj)
10     except JSONDecodeError, ValueError:
11         raise MalformedDataError('Issue JSON decoding config file')
12
13     if not isinstance(data, dict):
14         raise MalformedDataError('Must be a JSON-encoded dictionary')
15
16     if 'id' not in data:
17         return None
18
19     uid = data.get('id')
20     if isinstance(uid, str):
21         uid = int(uid)
22
23     assert isinstance(uid, int), 'The user id must be an integer'
24
25     return uid
```

MY FINAL VERSION

```
1 def get_id(conf_fname=CONFIG_FILE: str) -> Optional[int]:
2     """Return the user id from conf_fname, or None
3        if it hasn't been set yet
4
5        Precondition:
6            Valid JSON-encoded dictionary at conf_name
7     """
8
9     with open(conf_fname, 'r') as fobj:
10        data = json.loads(fobj)
11
12    uid = data.get('id')
13
14    assert uid is None or isinstance(uid, int)), \
15        'The user id must be an integer if it exists'
16
17    return uid
```

Self-renovation

RECOVERING FROM ERROR

WAYS TO DEAL WITH ERRORS

- Error codes (e.g. Go)
- Checked exceptions (e.g. Java)
- Unchecked exceptions (Python!)
- Abandonment (e.g. Erlang, Midori)

```go
type UserInfo struct {
    id int
}

func getId(configfile) {
    file, err := ioutil.
        ReadFile(configfile)
    if err != nil {
        return 0, err
    }

    var userinfo UserInfo

    err := json.Unmarshal(
        file, &userinfo)
    if err !=  nil {
        return 0, err
    }

    return userinfo.id, nil
}
```

GOLANG

JAVA

```java
public int get_id(String conf_fname)
      throws MalformedDataError {
  try {
    JSONParser jsonParser = new JSONParser();
    File file = new File(conf_fname);
    JSONObject config = jsonParser.parse(
                                new FileReader(file));
    parseJson(jsonObject);
    return object.id;
  } catch (FileNotFoundException |
           JsonDecodeException ex) {
    throw new MalformedDataError(ex);
  }
}
```

WHAT IS RECOVERABLE?

- Network flakiness
- Database out of connections
- Disk unavailable
- ~~Re-reading corrupt file~~

Well-isolated systems

With clear, enforceable interfaces/ contracts

HOW DO I RECOVER?

Abandonment isolates at the process level

- What level of isolation makes sense from you?
- How can you make sure all bad state is cleared away to retry?

LESSONS LEARNED

There are many kind of Exceptions in Python

- Control flow, bugs, contract failures

Make a contract

- Document the contract
- Use exceptions to attribute breaches in contract

Self-heal cleanly

- Think about what is recoverable
- Clear out bad state, get back to happy
- Isolated components are helpful

THANKS

ADDITIONAL READING

http://joeduffyblog.com/2016/02/07/the-error-model/

http://dafoster.net/articles/2016/05/17/abandonment-vs-unchecked-exceptions-for-error-handling/

http://cecs.wright.edu/~pmateti/Courses/7140/Lectures/OOD/meyer-design-by-contract-1992.pdf