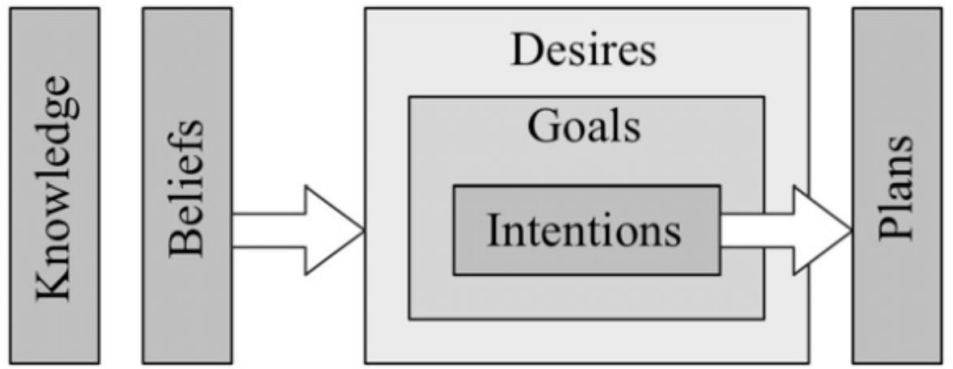
Intelligent Agents Using KQML and KIF

Individual Project

System Overview & Unit 6 Link

- Builds on the Unit 6 group design of autonomous data-retrieval agents.
- Guided by the **BDI model** of rational agency (Rao & Georgeff, 1995).
- Reflects current interest in **distributed, cooperative AI** (Xie et al., 2017).
- Emphasises **ethical AI** – transparency, fairness, accountability (Floridi & Cowls, 2019).
- Bridges theory and practice through a Python implementation of agent dialogue.



Source: Barba Guaman et al. (2017), BDI model figure

Architecture & Classes

- Object-oriented design using Python data classes
- Implements lightweight KQML message structure
- Modular, reusable, and testable agent components.

```python
from dataclasses import dataclass, field
from typing import Optional, Dict
import time, uuid

def _now_iso() -> str:
    return time.strftime("%Y-%m-%dT%H:%M:%S", time.gmtime())

@dataclass
class KQMLMessage:
    """
    Why this design:
    - Implements a lightweight version of the KQML (Knowledge Query and Manipulation Language).
    - Demonstrates how agents communicate using performatives and parameters.
    - Designed for clarity and explanation rather than complexity.
    """
    performative: str
    sender: str
    receiver: str
    language: str = "KIF"
    ontology: str = "research-ops-1"
    content: str = ""
    in_reply_to: Optional[str] = None
    reply_with: str = field(default_factory=lambda: f"msg-{uuid.uuid4().hex[:8]}")
    timestamp: str = field(default_factory=_now_iso)
    params: Dict[str, str] = field(default_factory=dict)
```

Agent Communication (KQML + KIF)

- Implements structured agent dialogue via KQML performatives (*ask-one*, *reply*)
- Messages contain metadata for sender, receiver, and ontology context
- Logical content expressed in KIF ensures interpretability and reasoning consistency
- Promotes transparency and traceability in autonomous communication (Finin et al., 1994; Genesereth and Fikes, 1992)
- Supports ethical accountability through explainable message structure (Floridi and Cowls, 2019)

```
# Initialise system
bus = MessageBus()
ext = Extractor()
ret = Retriever()
bus.register(ext)
bus.register(ret)

# Happy path
print("=== HAPPY PATH ===")
ret.start_dialogue("nano")

# Error path (empty query)
print("\n=== ERROR PATH ===")
ret.start_dialogue("")  # should produce an error
```

“Happy Path” Demonstration

- Demonstrates complete communication cycle between Retriever and Extractor.
- Valid data request processed and summarised as structured KIF output.
- Timestamped logs provide auditability and reproducibility
- Confirms successful task fulfilment and agent coordination (Jennings and Bussmann, 2003)
- Output illustrates system transparency and ethical clarity (Gal, 2022)

```
=== HAPPY PATH ===
SEND (ask-one
  :sender Retriever
  :receiver Extractor
  :language KIF
  :ontology research-ops-1
  :reply-with msg-286bcf7d
  :timestamp 2025-10-13T21:30:50
  :content (findPapers "nano"))

RECV (ask-one
  :sender Retriever
  :receiver Extractor
  :language KIF
  :ontology research-ops-1
  :reply-with msg-286bcf7d
  :timestamp 2025-10-13T21:30:50
  :content (findPapers "nano"))

RECV (reply
  :sender Extractor
  :receiver Retriever
  :language KIF
  :ontology research-ops-1
  :reply-with msg-60031a63
  :timestamp 2025-10-13T21:30:50
  :in-reply-to msg-286bcf7d
  :content ((Paper p1) (hasTitle p1 "Nanomaterials Survey") (hasDOI p1 "10.1234/nano.001") (hasSummary p1 "Overview of nanomaterial synt
hesis and properties.")))

RECV (reply
  :sender Extractor
  :receiver Retriever
  :language KIF
  :ontology research-ops-1
  :reply-with msg-60031a63
  :timestamp 2025-10-13T21:30:50
  :in-reply-to msg-286bcf7d
  :content ((Paper p1) (hasTitle p1 "Nanomaterials Survey") (hasDOI p1 "10.1234/nano.001") (hasSummary p1 "Overview of nanomaterial synt
hesis and properties.")))
```

Error Handling & Retry Logic

- Detects invalid or empty queries and returns structured error responses
- Prevents propagation of faulty requests across the network
- Logs each error with timestamps for traceability and recovery
- Implements retry mechanism for message resilience (Bellifemine et al., 2007)
- Meets ethical standard of explicability in AI system behaviour (Floridi and Cowls, 2019)

```
---- SUMMARY ----
- Nanomaterials Survey [10.1234/nano.001]
  Overview of nanomaterial synthesis and properties.

=== ERROR PATH ===
SEND (ask-one
  :sender Retriever
  :receiver Extractor
  :language KIF
  :ontology research-ops-1
  :reply-with msg-fecf2ee6
  :timestamp 2025-10-13T21:30:50
  :content (findPapers ""))

RECV (ask-one
  :sender Retriever
  :receiver Extractor
  :language KIF
  :ontology research-ops-1
  :reply-with msg-fecf2ee6
  :timestamp 2025-10-13T21:30:50
  :content (findPapers ""))

RECV (error
  :sender Extractor
  :receiver Retriever
  :language KIF
  :ontology research-ops-1
  :reply-with msg-aeb821c1
  :timestamp 2025-10-13T21:30:50
  :in-reply-to msg-fecf2ee6
  :content (reason EmptyQuery))

RECV (error
  :sender Extractor
  :receiver Retriever
  :language KIF
  :ontology research-ops-1
  :reply-with msg-aeb821c1
  :timestamp 2025-10-13T21:30:50
  :in-reply-to msg-fecf2ee6
```

Subscriptions & Reasoning

- Demonstrates *subscribe* performative for dynamic updates and ongoing monitoring
- Extractor sends *inform* messages autonomously when new data appears
- Introduces ranking and scoring of results to simulate relevance reasoning
- Embodies proactive and reactive agent traits (Wooldridge and Jennings, 1995)
- Enhances autonomy and adaptiveness within cooperative multi-agent systems (Nguyen et al., 2018)

```python
class Retriever(Agent):
    """Search & Retrieval with pending request map and correlation checks."""
    def __init__(self, name="Retriever"):
        super().__init__(name)
        self.last_answers: List[str] = []
        self.pending: Dict[str, str] = {}  # reply_with -> term

    def start_dialogue(self, term: str):
        ask = KQMLMessage(
            performative="ask-one", sender=self.name, receiver="Extractor",
            content=f'(findPapers "{term}")'
        )
        self.pending[ask.reply_with] = term
        self.send(ask)

    def subscribe_updates(self, term: str):
        sub = KQMLMessage(
            performative="subscribe", sender=self.name, receiver="Extractor",
            content=f'(findPapers "{term}")'
        )
        self.pending[sub.reply_with] = term
        self.send(sub)
```

Testing & Validation

- Automated tests executed using Python unittest framework
- Validates both successful communication and error recovery processes
- Confirms reliable message exchange under normal and fault conditions
- Reinforces system dependability and maintainability (Bellifemine et al., 2007)
- Transparent logging and reproducibility uphold ethical and technical standards (Floridi and Cowls, 2019).

```python
import unittest

class TestAgents(unittest.TestCase):
    def setUp(self):
        self.bus = MessageBus()
        self.ext = Extractor()
        self.ret = Retriever()
        self.bus.register(self.ext)
        self.bus.register(self.ret)

    def test_happy_path_returns_facts(self):
        self.ret.start_dialogue("nano")
        combined = " ".join(self.ret.last_answers)
        self.assertIn("hasDOI", combined)

    def test_error_on_empty_query(self):
        self.ret.start_dialogue("")
        self.assertTrue(any(ans.startswith("ERROR ") for ans in self.ret.last_answers))

suite = unittest.TestLoader().loadTestsFromTestCase(TestAgents)
unittest.TextTestRunner(verbosity=2).run(suite)
```

```
test_error_on_empty_query (__main__.TestAgents.test_error_on_empty_query) ... ok
test_happy_path_returns_facts (__main__.TestAgents.test_happy_path_returns_facts) ... ok

----------------------------------------------------------------------
Ran 2 tests in 0.012s

OK
```

Limitations & Future Work

- All design goals from Unit 6 successfully implemented in Python: data retrieval, processing, and structured KIF output.
- Agents demonstrated autonomy and cooperation via KQML performatives.
- Fault tolerance and retry logic improved robustness and reliability (Bellifemine et al., 2007).
- Communication remained interpretable, fulfilling ethical transparency standards (Floridi and Cowls, 2019).
- Code modularity and in-line documentation enhanced maintainability and scalability (Jennings and Bussmann, 2003).
- Minor limitations: static dataset and simplified ontology restrict external integration.

Conclusion & Ethical Reflection

- Developed a functional multi-agent system based on KQML and KIF protocols.
- Demonstrated autonomy, communication, and transparency.
- Fulfilled ethical design criteria through interpretability and accountability.
- Provides scalable foundation for responsible AI development.
- Future work: dynamic ontologies, API integration, and adaptive learning mechanisms.

References

References:

Barba-Guamán, L., Luján-Mora, S. and López, V. (2017) *Belief-Desire-Intention model for reasoning procedure*, in *A BDI Agent Model for Behavioural Change Process in People with Chronic Disease*. *ResearchGate*. Available at: https://www.researchgate.net/figure/Belief-Desire-Intention-model-for-reasoning-procedure_fig1_313617511 (Accessed: 12 October 2025).

Bellifemine, F., Caire, G. and Greenwood, D. (2007) *Developing Multi-Agent Systems with JADE*. Chichester: John Wiley & Sons. Available at: https://onlinelibrary.wiley.com/doi/book/10.1002/9780470058411 (Accessed: 12 October 2025).

da Costa, A.C.R., Lima, T., Feitosa, D. and Lima, R. (2020) *A Survey on Multi-Agent Systems: Concepts, Tools and Applications*. Engineering Applications of Artificial Intelligence, 90, p. 103509.

Finin, T., Fritzson, R., McKay, D. and McEntire, R. (1994) KQML as an Agent Communication Language.' Proceedings of the 3rd International Conference on Information and Knowledge Management. Available at: https://dl.acm.org/doi/10.1145/191246.191322 (Accessed: 12 October 2025).

Floridi, L. and Cowls, J. (2019) *Translating Principles into Practices of Digital Ethics: Five Risks of Being Unethical*. Philosophy & Technology, 33(4), pp. 701–722. Available at: https://link.springer.com/article/10.1007/s13347-019-00354-x (Accessed: 12 October 2025).

Gal, K. (2022) 'Multi-Agent Systems: Technical and Ethical Challenges.' Daedalus, 151(2), pp. 114–128. Available at: https://direct.mit.edu/daed/article/151/2/114/110611 (Accessed: 12 October 2025).

Genesereth, M.R. and Fikes, R.E. (1992) *Knowledge Interchange Format 3.0 Reference Manual*. Stanford University, Logic Group. Available at: http://www-ksl.stanford.edu/knowledge-sharing/kif/(Accessed: 12 October 2025).

Gil, Y., Greenspan, R., Krause, J. and Schulz, S. (2020) *Toward Trustworthy AI: Best Practices for Explainability and Testing*. AI Magazine, 41(3), pp. 10–23.

Jennings, N.R. (2001) *An Agent-Based Approach for Building Complex Software Systems*.Communications of the ACM, 44(4), pp. 35–41. Available at: https://doi.org/10.1145/367211.367250 (Accessed: 12 October 2025).

Jennings, N.R. and Bussmann, S. (2003) *Agent-Based Control Systems: Why Are They Suited to Engineering Complex Systems?* IEEE Control Systems Magazine, 23(3), pp. 61–74. Available at: https://doi.org/10.1109/MCS.2003.1200249 (Accessed: 12 October 2025).

Nguyen, D.T., Nguyen, T. and Nahavandi, S. (2018) *Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications*. IEEE Access, 6, pp. 28520–28544. Available at: https://ieeexplore.ieee.org/document/9043893 (Accessed: 12 October 2025).

Rao, A.S. and Georgeff, M.P. (1995) *BDI Agents: From Theory to Practice*. Proceedings of the 1st International Conference on Multi-Agent Systems (ICMAS '95). Available at: https://cdn.aaai.org/ICMAS/1995/ICMAS95-042.pdf (Accessed: 12 October 2025).

Shoham, Y. and Leyton-Brown, K. (2008) Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations. Cambridge: Cambridge University Press.

Wooldridge, M. and Jennings, N.R. (1995) *Intelligent Agents: Theory and Practice*. Knowledge Engineering Review, 10(2), pp. 115–152. Available at: https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated1/woodridge_intelligent_agents.pdf (Accessed: 12 October 2025).

Xie, J., Lui C.C.. (2017) Multi-Agent Systems and Their Applications. Artificial Intelligence Review, 47(3), pp. 1–33. Available at: https://www.researchgate.net/publication/318421603_Multi-agent_systems_and_their_applications (Accessed: 12 October 2025).