Botkit Conversational UI Design Spec

- Conversations designed as Python coroutines/generators
- `yield` statements indicate that you are waiting for a response from the peer
- Control flow is returned back to framework when a `yield` statement is used.

```
def ask_go_on_date_with_me(conv: Conversation):
    answer: Response = yield conv.ask("Would you like to go on a date with me?")

    if answer.intent == "affirmation":

        conv.say("wow... Okay, let me think about that...")
        yield conv.delay(typing=True, seconds=3)
        return conv.happy("Nice, alright! I'm in!")

    elif answer.intent == "negation":
        return conv.sad("You are a terrible person!")

    else:
        conv.angry("Hey! I asked you something!").continue_as_new()
```

- Dialogs can start other coroutines for the concept of **sub-dialogs**

```
@dialog(per_same_user=True)  # Every user gets their own state
def find_a_date(conv: Conversation, user_data: UserData):
    age: int = yield ask_age(conv)
    user_data["age"] = age

    user_data["gender"] = yield ask_boy_or_girl(conv)
    user_data["location"] = yield ask_location(conv)
```

- It is alright if those functions get big, as you should be able to quickly change them without much overhead and design the most engaging experience possible

```
def ask_age(conv: Conversation) -> Optional[int]:
    age_answer: Response = yield conv.ask("Soo, how old are you?",
force_reply=True)

    if age_entity := age_answer.entities.get("age"):
        age = int(age_entity)
    elif year_of_birth := age_answer.entities.get("year_of_birth")
```

```
        age = datetime.now().year - int(year_of_birth)
    else:
        conv.say("Oh, you don't wanna tell?").exit()
        return None

    if age < 20:
        return (
            conv.say(f"Ah, so nice that you're {age}!")
                .sad("But I was kinda looking for someone older...")
                .say("Maybe in a few years? :wink:")
                .exit("Goodbye!", stop_responding=True)  # (what a rude bot!)
        )

    return age


def ask_boy_or_girl(conv) -> Literal["gender_male", "gender_female",
"gender_other"]:
    # Choices get rendered as buttons automatically
    gender: Response = yield conv.ask(
        "Are you a boy or a girl?",
        choices=["A boy", "A girl", "Something else"]
    )
    return gender.intent
```

- Integrating Botkit Widgets into conversations

```
def ask_date_day_and_time(conv: Conversation, user_data: UserData) -> datetime:
    conv.say("So, when would you like to meet?")

    conv.invoke(DateTimePicker(
        min=datetime.utcnow(),
        timezone=user_data["timezone"],
        on_result_chosen="datetime_chosen"
    )

    while True:
        answer: Response = (yield)  # wait for events to happen
        if answer.event == "datetime_chosen":
            break

    return answer.event_payload
```

TODO

- [ ] Figure out how to persist coroutine state between restarts.
- [ ] It is unclear yet what the entry points to individual state machines should look like.