```
    template<typename UpdaterType>
    struct UpdaterTest : public UpdaterType
    {
        UpdaterTest() = default;

        void sendUpdate()
        {
            hasDelivered = false;
            UpdaterType::triggerAsyncUpdate();
        }

        void handleAsyncUpdate() override
        {
            hasDelivered = true;
            event.signal();
            JUCE_ASSERT_MESSAGE_THREAD;
        }

        WaitableEvent event;
        std::atomic<bool> hasDelivered { false };
    };
```

```cpp
template<typename UpdaterType>
struct UpdaterTest : public UpdaterType
{
    UpdaterTest() = default;

    void sendUpdate()
    {
        hasDelivered = false;
        UpdaterType::triggerAsyncUpdate();
    }

    void handleAsyncUpdate() override
    {
        hasDelivered = true;
        event.signal();
        JUCE_ASSERT_MESSAGE_THREAD;
    }

    WaitableEvent event;
    std::atomic<bool> hasDelivered { false };
};
```

```cpp
template<typename UpdaterType>
void runAsyncUpdateTest()
{
    UpdaterTest<UpdaterType> updater;
    PerformanceCounter pc ("RealTimeAsyncUpdaterCounter", 1000);
    std::atomic<bool> hasFinished { false };

    std::thread t ([&]
                   {
                       for (int i = 0; i < 10'000; ++i)
                       {
                           pc.start();
                           updater.sendUpdate();
                           updater.event.wait (-1);
                           pc.stop();

                           if (! updater.hasDelivered.load())
                               expect (false);
                       }

                       hasFinished = true;
                   });

    while (! hasFinished.load())
        MessageManager::getInstance()->runDispatchLoopUntil (5);

    t.join();
    expect (updater.hasDelivered.load());
}
```