Appendix A - Source documents

file1.html

```
<HTML>
<HEAD>
    <TITLE>Agent overview</TITLE>
    <META>Software agents, agent definition</META>
</HEAD>

<BODY>

<H1>What is an agent?</H1>
There are several interpretations of the meaning behind the term <I>agent</I>, and
therefore the term is not a strong one. Two main approaches attempt to define an
agent:
<OL TYpe="I">
    <LI> <B>Ascription</B> made of a person (what they are). An agent has different
         meanings for two people. Sometimes the agent-based approach fits the
         expectations of the programmer, sometimes it does not and should not be
         used. It should take the situation into account and provide feasibilities
         like reasoning.

    <LI> <B>Description</B> of the attributes of an agent (what they do).  A
        software entity that functions continuosly, autonomously and inhabitated by
        other agents. This means it should learn from experience and cooperate with
         his friends. Here you can find some <A HREF="file2.html">attributes</A> of
         an agent
</OL>


Do you wonder <A HREF="file3.html">why software agents</A> are appealing?
<A HREF="file4.html#pattie">Pattie Maes</A> is often debating agenthood.

</BODY>
</HTML>
```

file2.html

```
<HTML>
<HEAD>
    <META>Agent characteristics</META>
</HEAD>

<BODY>

<H1>Agent characteristics</H1>

The following <B>attributes</B> are common to find in an agent:
  <ul>
    <li> <I>Reactivity</I> - ability to sense and then act on its environment (it
         reacts on some stimuli it senses)
    <li> <I>Proactivity</I> - ability to start something itself, autonomously
    <li> <I>Collaboration</I> - work in concert with others
    <li> <I>Communication</I> with a person should be non-symbolic, but rather
         natural language-like
    <li> Use of <I>models</I> to infer new knowledge
    <li> <I>Continuity</I> persistent over time
    <li> <I>Adapting</I> to its environment, and learning from experience
    <li> <I>Mobility</I> - move itself from one place to another
  </ul>

<H1>Definitions</H1>

Gilbert uses a three-dimensional space to characterize agenthood:
  <ol style='margin-top:0cm' >
    <li> Degree of Agency - how autonomous is the agent?
    <li> Degree of Mobility - how much travel from machine to machine does the
         agent do?
    <li> Degree of Intelligence - Is reasoning and learning provided?
  </ol>

<P>Nana uses another classification resulting in four possible agent-types:
   Smart, Collaborative, Collaborative that learns and Interface-agents. </P>

<p>Dictionary: One that <A HREF="file3.html">acts on your behalf</A></p>



</BODY>
</HTML>
```

file3.html

```
<HTML>
<HEAD>

    <META>Software agents, direct manipulation interface agent</META>
</HEAD>

<BODY>

<H1>Why software agents?</H1>

Two motivations:
<OL>
    <LI> <B>Simplifying distributed computing</B>. Today's applications only
         cooperate in the most basic ways (file transfer, DB-queries etc). The web
         has evoluted from this basic communication to the
         <EM ONCLICK="ordlisteVindu('../ordliste.html#adhoc')" STYLE="cursor:hand">
         ad-hoc</EM> to the encapsulated message passing systems, all meaning low
         levels of interoperability. There is a need for <B>intelligent
        cooperation</B> among systems to optimize the work-processes towards goals.
         To increase the level of interoperability in small systems, an agent could
         serve as a global <B>resource manager</B>. For larger systems,  embedding
         peer-agents for each system may increase intelligence.
    <LI> <B>Overcoming user interface problems</B>. Direct manipulation has
         limitations like
</OL>


<TABLE border=1 CELLSPACING=2 WIDTH="90%" ALIGN="center">
<TR><TH>Limitations of direct manipulation</TH><TH>Advantages of agents</TH></TR>
  <TR>
  <TD>
    <ul><li>large spaces to be searched
        <li>difficult to schedule tasks
        <li>hard to make basic actions higher-level ones
        <li>consistency means predictable interfaces, but this is not so for complex
        tasks
        <li>software is function oriented rather than concerned with context of the
        task and situation
        <li>repetetive actions are not learned
    </ul>
  </TD>

  <TD>
    <ul><li>search and filtering mechanisms of the agent run in the background, help
            constrain the search space
        <li>event-driven actions/wake up on response
        <li>share our goals, they don't simply process our commands
        <li>may work around unforseen problems
        <li>account for context of the user's tasks and situation
        <li>learn from repetetive patterns
    </ul>
  </TD>
  </TR>
</TABLE>

As <A HREF="file4.html">debated</A> in the article "Direct Manipulation vs
Interface Agents" the two are complementary rather than mutually exclusive.
It is difficult to find a golden way between proactive and reactive behavior.

</BODY>
</HTML>
```

file4.html

```
<HTML>
<HEAD>
    <META>Ben Schneiderman Pattie Maes Debating</META>
</HEAD>

<BODY>

<H1>Debating...</H1>

Ben:
<UL>
    <LI> Anthropomorphic interfaces are not the future of computing.
    <LI> Great that Pattie is moving away from the "agent as living entity on
         screen"-vision
    <LI> Collaborative filtering will be important in the future.
    <LI> Adaptive features should appear as non-adaptation to the user, to be
         predictable. So adaptation must not lead to unpredictability.
    <LI> The user need to feel he did the job himself (not some magical agent)
    <LI> Words like smart, agent, intelligent etc mislead the designer to leave out
         important things in the user interface.
    <LI> A good thing to make the user model available for the user, but that is not
         being done today in most agent-systems
    <LI> Speech (NL) is not the future because it make use of the short-term memory
         and working memory. This degrades the level of performance. You do problem
         solving better when you use direct manipulation than speech.
    <LI> When it comes to the issue of critical time-restricted systems that should
        avoid mistakes, I think the essence is in designing a very simple interface
         (<A HREF="../341/foley.html">according to Foley</A>)
    <LI> Even blind people may use direct manipulation, because they are strong at
         spatial processing.
    <LI> Agent litterature does not focus enough on the user interface!
</UL>

<A NAME="pattie"></A>
Pattie:
<UL>
    <LI> Agents could work below the table, with a nice, possibly direct
         manipulation interface, that the user sees.
    <LI> Important to distinguish <A HREF="file3.html">software agents</A> from
         other agents.
    <LI> The disagreement is mainly due to us focusing on different problem domains.
         Ben looks at a structured task-domain with professional users, while I
         focus with end-users that are novices in a dynamic domain.
    <LI> Agree that speech is difficult. A lot of ambiguity has to be solved. But
        the agent-approach could use speech in <A HREF="cubricon.html">multilingual
         input-features</A>.
    <LI> It is difficult for an agent to always do the right thing, so therefore i
         have focused on areas where things need not be 100 % correct.
    <LI> As complexity increases, so does the need for delegation.
</UL>

</BODY>
</HTML>
```