



Development Principles

- ♦ Highly distributed, loosely organized – this allows nodes to come and go, be added or removed without adversely affecting the system as a whole. This also adds to the robustness of the system – no single points of failure or attack.

- ♦ Service Oriented Architecture (SOA; machine-to-machine automation) – highly interfaced for extensible, collaborative development and maintenance. Well defined interfaces between modules and elements – coding becomes less important and more flexible and sustainable. User interfaces just become additional service calls to the system, but now systems can be automated since all functions are service calls that can be programmed.

- ♦ Modular, decoupled functionality – functions are separated into distinct modules with well defined interfaces which allows better design, re-factoring, and extensibility.

- ♦ Scalable – grows without systemic alteration

- ♦ Extensible – altered without systemic alteration

- ♦ Pipeline – flexible, content agnostic. This can also be described as data-centric design. The system acts more as a pipeline – moving data through the system having specific understanding of the data. Data is used to define data (meta-data);

- ♦ Web launched Rich Internet Application (RIA) – as available as common browser based interfaces, but richer in capabilities since they are not constrained by the browser’s capabilities.

- ♦ Use open source code when possible – leverage existing code and frameworks where it makes sense and does not create licensing issues.

- ♦ Cross platform code foundation (e.g. Java, Python, PHP, etc.)

- ♦ Semantic messaging (e.g. XML) – This is the implementing method for the system’s distributed, loosely coupled SOA. State and commands are communicated via messages that have semantic data.