Intelligent agent technology is at an intriguing stage in its development. Commercial strength agent applications are increasingly being developed in domains as diverse as manufacturing (Deen, 2003; Bussman et al., 2004; Jarvis et al., 2008a) war gaming (Jones et al., 1999; Heinze et al., 2002) and UAV mission management (Karim and Heinze, 2005). Furthermore industrial strength development environments are available, e.g. (AOS Group, 2012; University of Michigan, 2012; JADE, 2012) and design methodologies (Padgham and Winikoff, 2004) reference architectures (van Brussel et al., 1998) and standards (IEEE Computer Society, 2012) are beginning to appear. These are all strong indicators of a maturing technology. Furthermore, it has been proposed as the paradigm of choice for the development of complex distributed systems (Decker, 2004) and as the next step forward from object oriented programming (Wooldridge and Jennings, 1995). However, the uptake of the technology is not as rapid or as pervasive as its advocates anticipated.

In this chapter, we begin by looking at where multi-agent systems technology is positioned in terms of its commercial uptake. We will then provide an overview of the BDI model of agency which underpins GORITE and finally, we will introduce GORITE and the concept of team programming. It is our belief that team programming, with its twin focus on organizational structure and coordinated behaviour may offer a way forward for the multi-agent systems paradigm.

1.1 Two “Success” Stories

In this section, our discussion will focus on two areas – one, a niche area where multi-agents system have achieved commercial success (computer generated forces) and the other where extensive research activity over many years has not yet resulted in significant industry uptake of the technology (manufacturing). We will then consider what these two very different stories might hold for the future of the technology.

In the military domain computer generated forces (CGFs) are widely employed in both training and research. Systems such as VBS2 (Bohemia Interactive, 2012) provide a rich palette of entities (agents) at various organizational levels (e.g. soldiers, sections, platoons, companies) and a wide range of behaviours at varying degrees of autonomy (e.g. player controlled movement, inbuilt path-finding). Scenarios for training or research purposes can be constructed and played out
using built in, scripted or player provided behaviours as appropriate. Depending
on the scenario, complex interplays involving autonomous groups of entities may
be required. For example, a section may be required to retain formation while
moving through difficult terrain or while they are under fire. Also, a company
attack will require the coordination of fire platoon and attack platoons (Connell
et al., 2003).

With VBS2 and its ilk, the norm is to provide access to the internal behaviour
of these systems through mechanisms such as APIs, scripting languages or TCP/IP
messaging. As a result, external processes are often able to control and augment
the behaviour of CGF entities, in particular customizing entity behaviours to align
with local military doctrine and terrain. The multi-agent paradigm has achieved
considerable commercial success in this regard by using the CGF platform as
essentially a visualization vehicle for individual entities. Entity behaviour,
organizational structures and organizational behaviour are then implemented
externally to the CGF platform in a separate multi-agent based application which
interacts with the CGF platform through message passing (Lui et al., 2002;
Connell et al., 2003; Jarvis et al., 2005).

The traditional use of computer generated forces has been at the theatre level –
military resources move (or are moved) on a map of the area of engagement, as
illustrated in Figure 1.1. However, given the continuing decrease in the
cost/performance ratio in games technology, video games are becoming increasingly
deployed in military training, as evidenced by the success of America’s Army (U.S.
Army, 2012). The success of this approach is critically dependent on the quality of
the behaviour provided by the NPCs (non-playing characters) in the game. In
conventional gaming, acceptable NPC behaviour has usually been able to be realized
with simple techniques, such as state machines and flocking algorithms (Millington
and Funge, 2009). However, military training imposes much more stringent
realism requirements on NPC behaviour and multi-agent systems are being used to
provide this realism. Recent examples of this include requirements for realistic
group behaviour of NPC targets when fired upon (Jarvis et al., 2004), the
incorporation of fatigue and fear into NPC behaviour (Evertsz et al. 2007) and the
extension of military doctrine to incorporate legal and political considerations
(Evertsz et al. 2009).

Likewise, the manufacturing domain has provided a similarly rich and varied
playpen for the exploration of the multi-agent paradigm. However, while military
multi-agent systems activity has focused on virtual applications such as
simulation-based training, the focus for manufacturing has been the management
of manufacturing activity. Despite significant industrial involvement in multi-
agent systems research (e.g. the Holonic Manufacturing Systems (HMS) project
(Brennan and Norrie, 2003)), the technology has not yet established a foothold in
manufacturing. While there is a widely held belief that the key to realizing
effective utilisation of resources in a dynamic, market-driven manufacturing
environment lies in the adoption of the multi-agent paradigm, that belief has not
been evidenced in practice. We suspect that the reason for this lies with the
layered structure of manufacturing operations. Traditionally, manufacturing
operations have been partitioned into three layers – planning, scheduling and